PIKA Application Development Suite (PADS) 2.0

User Guide

Table of Contents

1 Copyright Information

2 Contacting PIKA Technologies

Customer Care

PIKA Technologies provides free technical support to all customers. For support issues, phone or e-mail our Customer Care department at the following:

Tel: +1-613-591-1555

FAX: +1-613-591-9295

Email: support@pikatech.com

International Headquarters

PIKA Technologies Inc.
535 Legget Drive, Suite 400
Ottawa, Ontario, Canada K2K 3B8

Tel: +1-613-591-1555

FAX: +1-613-591-9295

Email: sales@pikatech.com

Internet

Visit our website at **www.pikatechnologies.com** for the latest news, product announcements, downloads, online community, documentation updates, and contact information.

3 Introduction

The PIKA Application Development Suite (PADS) is a development environment that allows users to create applications for the PIKA WARP the Appliance.

Guide Organization:

- Introduction - Describes the purpose and scope of the guide, and references to related documents.
- **Embedded Systems Overview (pg. 5)** - High level description of embedded system concepts
- **PADS Overview (pg. 6)** - High-level overview of PADS
- **Getting Started with PADS (pg. 8)** - Describes how to set up your development system and build and run your first software load for the appliance.
- **Software Package Information (pg. 18)** - Describes the packages available through PADS
- **Navigating the PADS Menu (pg. 36)** - Describes how to use the PADS menu to select packages
- **Developing Software for the Appliance (pg. 43)** - Design guidelines for the appliance and how to build your application using PADS
- **Adding a Package to PADS (pg. 51)** - Describes how to use PADS to cross-compile your application and make it available from the menu
- **Using PADS Rules to Build Software (pg. 63)** - Describes alternative commands to use when building software using PADS
- **Using Flash Memory to Run Your Application (pg. 65)** - Describes the flash memory on the appliance and how to update the software in flash
- **Advanced Topics (pg. 74)** - Technical details about the appliance and additional package development information
- **Frequently Asked Questions (pg. 87)** - Answers to typical users' questions
- **Troubleshooting (pg. 88)** - Typical problems and their solutions
- **Appendix A - LCD API Reference (pg. 91)** - API reference for updating the appliance LCD display
- **Appendix B - Timezone Codes (pg. 115)**

3.1 Purpose and Scope

The PIKA Application Development Suite (PADS) is the software component of PIKA WARP the Appliance. It offers developers the ability to add and modify components to provide value-added features when deploying customized versions of the appliance. PADS can be used on any Linux distribution and includes all the components necessary to successfully cross-compile applications and build software images for the appliance.

This guide describes how to develop custom applications for the appliance and how to use PADS to build and run the software.

3.2 Assumed Knowledge

We assume you have the following knowledge:

- Linux operating system
- Makefiles
- gcc development suite
- Asterisk knowledge to use the appliance as an Asterisk PBX
- telephony concepts to create telephony applications for non-Asterisk systems

3.3 Related Documents

The following documents are related to the PADS User Manual. These documents are linked together and constitute the complete set of documentation for the appliance. All documents are available at **http://www.pikatechnologies.com/appliancedownloads**.

- **PIKA WARP the Appliance User Guide**: This guide describes installation and configuration of the appliance.
- **PIKA WARP the Appliance Hardware Manual**: This manual describes the appliance base board and plug-in modules.
- **PIKA WARP the Appliance Release Notes** - These notes describe the contents of the release, including known product issues.

4 Embedded Systems Overview

The appliance is an embedded system designed to function as a small IP/Analog/Digital PBX or to run small computer telephony applications. This section describes some embedded system concepts you will need to understand in order to develop software for the appliance.

An embedded system is a combination of computer circuitry and software designed to perform a narrow range of pre-defined tasks, as opposed to a general-purpose computer which is intended to perform multiple tasks. Embedded systems do not usually have any of the typical computer peripheral devices such as a keyboard, display monitor, mass storage (e.g., hard disk drives), etc. or any kind of user interface software. This can make it possible to greatly reduce the complexity, size and cost as well as increase the robustness of embedded systems as compared with general-purpose systems. The lack of peripheral devices and narrow range of functions can also contribute to a lower power consumption. Embedded systems are often required to provide real-time response.

In contrast to general purpose computers, for which very few processor architectures are used (mostly the x86), embedded systems typically utilize numerous, competing processor architectures (PowerPC, ARM, etc.).

The software written for embedded systems may be referred to as firmware, and is stored in read-only memory or flash memory chips rather than a disk drive. It often runs with limited computer hardware resources (processing power, memory). Some embedded systems include an operating system, which is referred to as an embedded operating system. It can be a very small operating system that was developed specifically for use with embedded systems, or it can be a stripped down version of a system that is commonly used on general-purpose computers, such as Linux.

Development for embedded systems is done on a separate computer because the embedded platform does not have the resources (hardware or software) to support compiling and linking programs. The computer used for development is typically a desktop PC and because it usually uses a different processor than the embedded (target) system, the process of producing machine code for a different processor is referred to as cross-compiling.

5 PADS Overview

PADS is designed to provide a user-friendly, open source framework to allow developers to easily create custom applications for the appliance. PADS simplifies embedded development by hiding the more complex aspects of embedded tool kits. This section describes the high-level concepts of the PADS framework.

PADS is based on the "Buildroot" framework. Buildroot is a set of Makefiles and patches that makes it easy to generate a cross-compilation tool chain and root file system for a target Linux system using the uClibc library. Buildroot is useful mainly for small or embedded systems. Embedded systems often use processors that are not the regular x86 processors used on a typical PC, such as PowerPC which is used on the appliance.

PADS is also a package selection framework. It allows users to select from a set of packages, each of which provides a framework to build a self-contained piece of functionality from source code. The set of packages provided in PADS includes those developed by PIKA plus third-party open source packages selected by PIKA to provide additional useful functions for the appliance. Additional packages may be added by developers. Package selection is controlled by a menu system. The packages selected determine the software capabilities of the appliance.

Each package has its own configuration settings and makefile. Package configuration includes the menu settings, whether the package is part of the default configuration plus any functional dependencies on other packages. The makefile defines the package version, how the package source is obtained and how to build the package which typically includes settings to cross-compile the package source code for the appliance target architecture.

The mechanism used to obtain the source code is package-specific and is determined by code owners who make their open source software available. PADS supports two mechanisms to retrieve package source code:

1. As tarballs from third-party sites or PIKA's FTP site
2. From SVN repositories, either a third-party repository or PIKA's SVN repository.

Refer to **Software Package Information (pg. 18)** for descriptions of the individual packages that PIKA makes available.

PADS provides the ability to build software images that can be stored in the flash memory. Images are a collection of software programs combined into a single binary file. The following images can be created using PADS:

- U-Boot (bootloader)
- Kernel
- Ramdisk
- Persistent file systems

Program	Description
bootloader	The bootloader is the program responsible for: • configuring the FPGA • performing basic hardware validation • loading the operating system, also referred to as the kernel, into memory It then transfers control to the operating system to continue processing. The bootloader used by the appliance is called U-boot.
kernel	The kernel is responsible for initializing the hardware and loading the main program into memory. The appliance uses a modified version of the open source Linux kernel version 2.6.26.5. PIKA has modified the code for use with the appliance.
ramdisk	Software ramdisks use the main memory as if it were a partition on a hard drive. The appliance ramdisk contains an ext2fs-based temporary file system using a standard Linux file system layout which is loaded into RAM at boot time. All the programs that run on the appliance (e.g. libraries, applications, Asterisk) are located in this temporary file system. Changes to this file system are temporary and are lost the next time the appliance is rebooted. The appliance uses a special section of the flash memory formatted as a Journaling Flash File System, version 2 (JFFS2) for persistent data, such as configuration information.

Refer to section **Using Flash Memory to Run Your Application (pg. 65)** for information about writing images into flash memory.

6 Getting Started with PADS

The following sections describe how to:

- set up your development computer to use PADS
- build software to run on the appliance with the default packages selected in PADS
- run the new software on the appliance using NFS

6.1 Development System Setup and Configuration

A separate Linux system is required to use PADS to cross-compile applications and to run software on the appliance. The following sections describe the steps to set up your Linux development computer to use PADS.

6.1.1 System Requirements

Your development computer requires the following Linux packages in order to use PADS:

- A serial client (e.g. minicom on Linux or HyperTerminal on Windows)
- TFTP (Trival File Transfer Protocol) Server
- NFS (Network File System) Server
- WGET
- Subversion (SVN) 1.4 or greater (CentOS 4/RedHat 4 will likely need an update to obtain a newer version)
- MAKE
- AUTOCONF
- AUTOMAKE
- LIBTOOL
- NCURSES 5.4 or greater
- NCURSES-DEVEL 5.4 or greater
- PATCH
- PATCHUTILS
- SSH client
- GCC 4.x or greater (CentOS 4/RedHat 4 will likely need an update to obtain a newer version)
- module-init-tools 3.2 or later (CentOS 4/RedHat 4 will likely need an update to obtain a newer version)

SELinux and any firewall software must be disabled on your development computer to run TFTP and NFS.

6.1.2 Setting up TFTP and NFS

Setting Up TFTP

A TFTP server must be running on your development computer to load software on to the appliance over the network. Software accessed via TFTP is loaded into memory before it is copied into flash memory.

TFTP is controlled by the extended Internet services daemon (xinetd), which is responsible for starting services related to Internet access. The xinetd package must be installed to use TFTP. If your Linux distribution supports a package manager such as rpm, yum (Red Hat-based systems) or apt (Debian), use it to install the xinetd and tftp server packages. If your Linux distribution does not support a package manager, obtain the source for the xinetd and tftp packages, compile and install them on your system.

In the directory /etc/xinetd.d, ensure that there is a configuration file called tftp containing the following settings:

```
service tftp
{
        disable = no
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

Start the xinetd and tftp services using the appropriate command for your Linux distribution. Both the xinetd and tftp services should be included in the list of services to start at boot time using the appropriate mechanism for your Linux distribution (e.g. chkconfig, update-rc.d or rc-update).

Create a directory called /tftpboot on your development computer. If you do not want to create the

directory at the root of the file system, you can create a directory elsewhere and create a symbolic link to /tftpboot. To give all users full permissions, use the following command:

```
chmod a+rwx /tftpboot
```

To check if tftp is running, use the netstat command:

```
netstat -a | grep tftp
```

should return:

```
udp 0 0 *:tftp *:*
```

Refer to the following websites for more information:

- **Installing TFTP** - Describes installing the TFTP server software using various package managers and configuring TFTP for various Linux distributions
- **Installing Linux Software** - Describes using various package managers and installing from source

Setting up NFS

The appliance can run software from flash memory or via a network file system (NFS) located on your development computer. The section **Running the Software from NFS (pg. 13)** explains how to use NFS to run the software.

Ensure that the NFS service is installed on your development computer using the appropriate package installation mechanism for your Linux distribution.

Specify your NFS path by adding the following line in the file /etc/exports (you may need to create the file). <Your PADS path> is the location of your PADS source code (refer to section **Building Software for the Appliance (pg. 12)** for more information).

```
<Your PADS path>/build_warp/root *(rw,no_root_squash,no_subtree_check)
```

Using the appropriate commands for your Linux distribution, ensure that the nfs service is running and set to start at boot time.

NOTE:	If you change <Your PADS path> and you want to use the new directory for NFS, you must change the line in /etc/exports and run "exportfs -a" to export the new file system path.

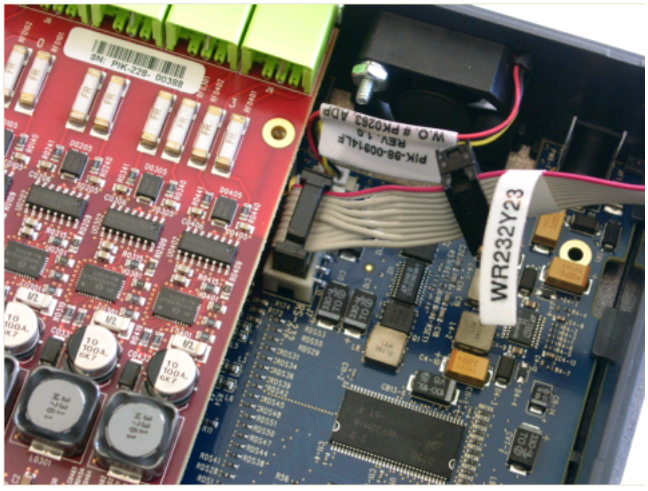
6.1.3 Configuring Serial Access

A serial connection to the appliance is required to access the boot loader console in order to set the boot loader environment variables and to view any output from the boot sequence or run-time output from applications. The sections below describe the required steps to connect your development computer to the appliance using the serial cable provided with the appliance development kit and how to configure your serial client.

Connecting the Serial Cable

Ensure that the appliance is powered off. Remove the screws on each of the side panels on the appliance.

Remove the top cover. You must be grounded with an anti-static wrist strap. Plug in the end of the serial cable provided with the appliance to the connector as shown below. The cable can only be plugged in to the connector using the correct orientation. Connect the other end to the serial port on your development computer.



Configuring a Serial Client

A serial client, such as minicom, is required to access the serial console. Use the following settings for the serial client:

Speed	115200
Parity	None
Bits	8
Stop Bits	1
Hardware Flow Control:	No
Software Flow Control:	No

Refer to the following web pages for additional information:

- **Setting up minicom** - Describes how to set up minicom for a Linux system
- **Setting up terminal programs** - Describes how to set up various serial clients

6.2 Building Software for the Appliance

Obtaining PADS

PADS can be obtained either as a tarball downloaded from the **http://www.pikatechnologies.com/appliancedownloads** or from PIKA's SVN repository (http://svn.pikatech.com/pads/distro). The SVN path for the latest PADS release is available on the PIKA Technologies website. For information about using SVN, refer to the Subversion website: **subversion.tigris.org**.

In the following sections, <Your PADS path> refers to the directory into which you unpacked the tarball or the directory specified for your checkout from SVN. The commands specified in the following instructions should be executed your Linux development computer. **You must run as root to build software using PADS.**

In the following examples, the local copy of PADS will be in a directory called PADS_2.0.6.x, where x is the build number.

Tarball from the Website:

```
tar -zxvf PADS_2.0.6.x.tgz
```

From the SVN Repository:

```
svn checkout http://svn.pikatech.com/pads/distro/tags/2.0.6.x PADS_2.0.6.x
```

Building the Software

In the directory <Your PADS path>, enter the command:

```
make menuconfig
```

This command displays the package selection menu. For now, you will use the default menu selections. Refer to **Software Packages Available in PADS (pg. 18)** for a description of the all the packages and to **Navigating the PADS Menu (pg. 36)** to learn about using the menu system to select individual packages. Use the arrow keys to select "Exit". Select 'Yes' when asked if you want to save your configuration.

Once you have exited the menu, enter the command:

```
make
```

This will build the software for the default packages. When the build is complete, you will have an NFS mount point at <Your PADS path>/build_warp/root.

6.3 Running Software from NFS

This section assumes that:

- you have installed the appliance according to the instructions in the **Getting Started - Hardware Installation** section of the PIKA Warp the Appliance User Guide
- you are connected to the appliance using the serial port
- your serial client is running.

It is expected that NFS will be the primary method for running software on the appliance during development. It is faster to boot using NFS, updates to files can be done without taking the time to write new images into flash and, depending on the file type being modified, without rebooting. The kernel, ramdisk and persistent file systems can be access from your development computer through NFS.

		Description
Kernel	Copy the file cuImage.warp from <Your PADS path>/images to /tftpboot on your development machine	• This image file was created when you built the software using the instructions in the previous section. • Ensure that TFTP is running on your development computer.
Ramdisk	The mount point is <Your PADS Path>/build_warp/root	• The mount point was created when you built the software using the instructions in the previous section. • Ensure that the path in /etc/exports on your development computer matches this path. • Ensure that NFS is running on your computer.
Persistent file system	<Your PADS Path>/build_warp/root/persistent	• To modify persistent data, files may be updated on your development computer and will be reflected on the appliance immediately. • Services that use these files may need to be restarted (e.g. Asterisk).

If the path used for NFS is deleted, the appliance will no longer function. Any attempts to perform operations will return either "Unknown command" or "Stale NFS file handle". The only way to recover is to press the reset button to reboot. An NFS mount point must be rebuilt before booting. Simply rebuilding the software at the mount point will not recover the system; the system must be rebooted.

Any errors when booting using NFS will be shown on the serial display. Refer to **Troubleshooting (pg. 88)** for more information.

Setting up U-Boot

You must now configure the bootloader (U-Boot) on the appliance. Press the reset button on the appliance and then press any key when the boot sequence is shown on the serial client. The following shows the first part of the boot sequence. You must press a key after this part of the boot sequence:

```
U-Boot 1.3.0-62 (Oct 4 2008 - 12:13:33)

CPU:   AMCC PowerPC 440EP Rev. C at 533.333 MHz (PLB=133, OPB=66, EBC=66 MHz)
       I2C boot EEPROM enabled
       Bootstrap Option H - Boot ROM Location I2C (Addr 0x52)
```

```
, PCI async ext clock used       32 kB I-Cache 32 kB D-Cache


  Board: PIKA Embedded Appliance

I2C:   ready
DRAM:  256 MB
### Press 'p' to enter POST ###:  0
FLASH:  4 MB
NAND:  256 MiB
In:    serial
Out:   serial
Err:   serial
Protected 4 sectors


FPGA download...complete.

FPGA code revision 1.3.3.8

Net:   ppc_4xx_eth0
..ENET Speed is 100 Mbps - FULL duplex connection (EMAC0)
```

but before the end of countdown in this part of the boot sequence:

```
Hit any key to stop autoboot:  3
```

You will see the U-Boot prompt which is indicated by '=>'. The following commands are used when changing the U-Boot environment variables:

U-Boot Command	Purpose
setenv	Sets the environment variable to a specified value • Syntax: setenv <environment variable name> <environment variable value>
printenv	Prints the list of U-Boot environment variables
saveenv	Saves changes to U-Boot environment variables

Set the following variables using "setenv". When you are finished, enter the command "saveenv".

Environment variable changes do not take effect until the system is rebooted, either by pressing the reset button or by entering the "reset" command at the U-Boot prompt.

U-Boot Environment Variable Name	Value
serverip	The IP address of your development computer
ipaddr	The IP address of the appliance, it must be set to an IP address that is valid on your network. The factory preset value is "deflt". Note that this will override the IP address in the file /etc/networking.conf. Refer to **Network Settings (pg. 79)** for more information about network information settings.
gatewayip	The IP address of your network gateway, the default is 0.0.0.0
netmask	The netmask of your network, the default is 255.255.255.0.
rootpath	The directory where you have compiled the appliance code: <Your PADS path>/build_warp/root. Ensure that the path specified matches the path in the file /etc/exports (described in **Setting up TFTP and NFS (pg. 9)**).
bootcmd	run net_nfs • Instructs the bootloader to load the kernel via TFTP and to run the ramdisk from the NFS mount point specified in the ***rootpath*** environment variable. By default, it is set to "run nand_boot" which boots both the kernel and ramdisk from flash memory.

Refer to **U-Boot Environment Variables (pg. 83)** for a full list and description of all the U-Boot environment variables.

6

To change U-Boot settings, enter the command "reboot" from the Linux prompt on the appliance or press the reset button on the appliance and then interrupt the boot process by pressing any key as described above to get back to the U-Boot prompt. U-Boot settings cannot be changed while the system is running.

6.4 Logging in to the Appliance

When the appliance is booted, you will be presented with the login prompt on the serial console. To login, use the following:

- Userid: root
- Password: pikapika

To change the root password, enter the following at the Linux command prompt **on the appliance**:

```
passwd root
```

Follow the prompts to enter and confirm the new password. Password information is preserved across reboots.

An SSH service is included in the default software on the appliance and can also be used to login to the appliance. Ensure that you have configured the appliance for network access using the instructions in **Network Setup** in the Appliance User Guide for more information.

6.5 Making a First Asterisk Call

To verify that your software is functioning correctly, you can use Asterisk to make a test call. Plug a standard phone set into the built-in FXS port and dial extension 600. This will connect you to the Asterisk echo test recording.

Refer to **Making Asterisk Calls** in the PIKA WARP the Appliance User Guide for other Asterisk extensions available for the appliance.

7 Software Package Information

This section describes the software packages that are available in PADS, including the package menu name, the package directory name (relative to <Your PADS path>/package) and important information about using the package.

For information about selecting packages using PADS, refer to the section **Navigating the PADS Menu (pg. 36)**.

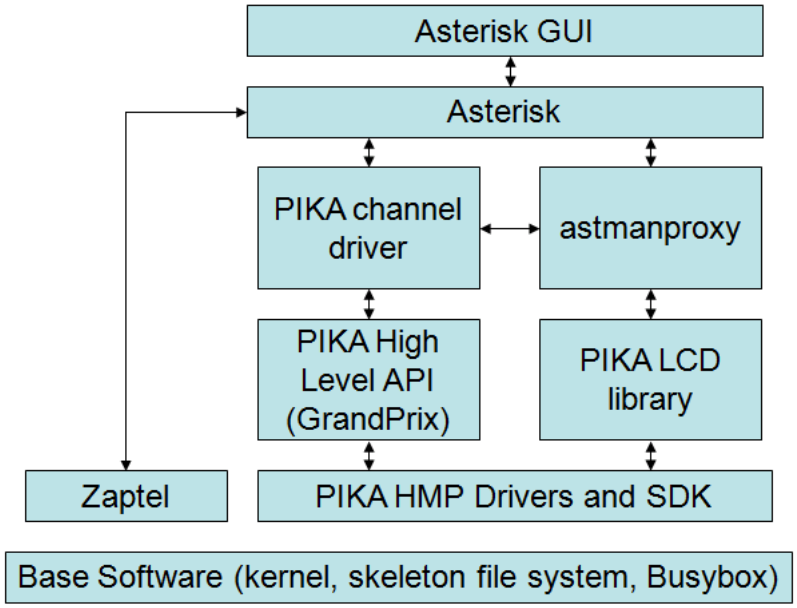
The types of packages available in PADS are:

- packages developed by PIKA Technologies to support the appliance hardware or to facilitate telephony application development on the appliance
- third-party packages chosen to add flexibility to the capabilities of the appliance, including those related to Asterisk PBX functionality

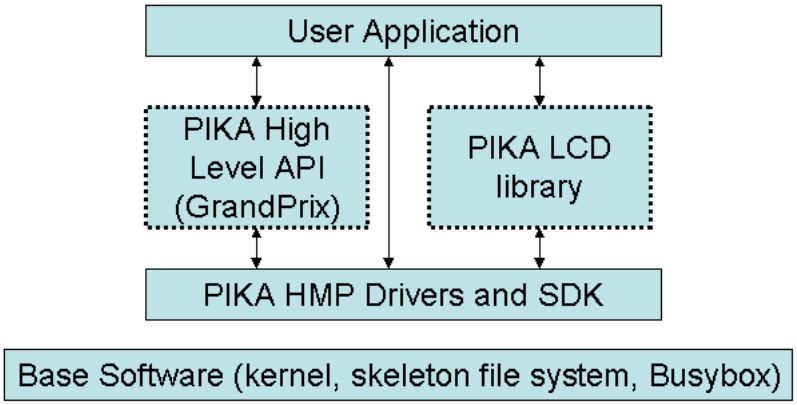
NOTE:	Third-party packages are available through PADS for convenience only and are provided "as-is". **PIKA does not support third-party packages**. Issues and questions about these packages should be directed to the package owners indicated in the package descriptions.

The factory default software on the appliance enables it to run as an Asterisk PBX. The following diagram shows the main software included. Various other utilities are also included:

- **Network Applications (pg. 29)**
  - ntpd
  - tftp server
  - http server
  - ssh server
- timezone utility - **Timezone (pg. 28)**
- warploader - **Software Update Utilities (pg. 33)**
- **ext2 File System Utilities for USB and SD Media (pg. 32)**



The following diagram shows the packages required to build telephony applications using the appliance hardware. The LCD library is optional if the application does not need to write to the LCD. Use of the High Level API is optional if the user application needs fine-grained hardware and software control. Note that the application may use both the High Level API as well as bypass mode to access the HMP SDK. For more information, refer to the GrandPrix and HMP documentation which is available at **www.pikatechnologies.com**.



7.1 Base Software

The appliance base software is composed of the following packages:

- **Linux kernel (pg. 20)**
- **"skeleton" file system (pg. 20)**
- **BusyBox (pg. 20)**
- **daemontools (pg. 21)** for Linux service management

7.1.1 Linux Kernel

The appliance uses a customized version of the standard Linux kernel, version 2.6.26.5. Additional features can be added using the **Kernel Configuration Menu (pg. 37)**.
The kernel package directory is <Your PADS path>/package/linux.

7.1.2 Skeleton File System

The package referred to as the "skeleton" is the base for the ext2 file system that resides in RAM at run-time. It is laid out in a typical Linux file system structure. The skeleton includes cross-compiled versions of standard library files as well as device nodes.

The skeleton package directory, <Your PADS path>/package/skeleton, contains system configuration files and scripts (including those required for initialization, refer to **Initialization Sequence (pg. 45)** for information) to configure the appliance for proper operation. These files are copied into place at build time.

7.1.3 BusyBox

BusyBox combines tiny versions of many common UNIX utilities into a single small executable. It provides minimalist replacements for most of the utilities usually found in GNU coreutils, util-linux, etc. The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts.

BusyBox has been written with size-optimization and limited resources in mind. It is also extremely modular so you can easily include or exclude commands (or features) at compile time. This makes it easy to customize your embedded systems. BusyBox provides a fairly complete POSIX environment for any small or embedded system. For more information or Busybox support, consult **www.busybox.net**.

Refer to section **Busybox Configuration Menu (pg. 38)** for information about selecting additional BusyBox components.

The BusyBox package directory is <Your PADS path>/package/busybox.

7.1.4 daemontools

The daemontools software is included on the appliance. It provides the ability to start and monitor services. The tools can do the following:

- restart a service if it dies
- provide information about the service status and the length of time the service has been running
- log service error messages

The following services are under the control of the daemontools:

- Asterisk
- autoflash-sd
- autoflash-usb
- dropbear
- ntpd

The following table lists some of the commonly used daemontools commands.

Command	Purpose
svstat /service/<service name>	Indicates whether the service is up and if so, how many seconds it has been up
svc -u /service/<service name>	Starts a service, if it dies, it will be restarted
svc -d /service/<service name>	Stops a service and does not restart it after it stops
svc -o /service/<service name>	Starts a service once but will not restart it if it dies
svc -k /service/<service name>	Kills a service

For information and support, refer to **cr.yp.to/daemontools.html**

7.2 PIKA Drivers and SDKs

PIKA provides drivers and SDKs to enable user applications to control the appliance hardware and to write telephony applications.

Package	Package Directory Name	Dependencies	Description
HMP Low-Level API version 2.7.x	hmp	none	This package provides driver support for the appliance telephone interfaces (FXS, FXO and BRI ports), the audio ports and the LCD. PIKA HMP also provides low-level API for developing telephony applications including VoIP, and voice processing such as record, play, DTMF detection, tone detection and generation, and voice detection. For more information about using PIKA HMP to build telephony applications, refer to the HMP documentation which is available at **www.pikatechnologies.com**. The HMP package is required for telephony applications with or without Asterisk.

High Level API version 2.7.x	grandprix	HMP	PIKA GrandPrix (GP) is a software layer on top of the HMP low-level API that makes it easier and faster for designers to develop user applications based on PIKA hardware and software. It removes most of the in-depth knowledge required to develop user applications to make calls using PIKA hardware; play and record files; and perform media analysis such as digit and tone detection, call progress, and call analysis. At the same time, it has the flexibility to co-exist with the low-level API. For more information about using GP to develop telephony applications, refer to the GP documentation which is available at **www.pikatechnologies.com**. The PIKA Channel Driver for Asterisk (described in section **Asterisk (pg. 23)**) includes its own copy of PIKA Grandprix. Choose GrandPrix if you want to develop non-Asterisk telephony applications using the appliance hardware and the GrandPrix high-level API.
LCD Library version 1.0.x	lcdlib	HMP	An application can use this library to display information on the appliance LCD. Asterisk uses it to display the call status. If you have a custom application that writes to the LCD, the astmanproxy package (described in section **Asterisk (pg. 23)**) should not be selected, as the applications will overwrite each other's data. For information about using the API, refer to section **Displaying Information on the LCD (pg. 80)**.

7.3 Asterisk and Related Packages

The following packages are required to use the appliance as an Asterisk PBX. All packages are included in the default software for the appliance.

Component	Package Directory Name	Dependencies	Description

Asterisk version 1.4.21.2	asterisk	none	Asterisk is an open source PBX and can be used on the appliance as the core of an IP or hybrid PBX, switching calls, managing routes, enabling features, and connecting callers with the outside world over IP, BRI and POTS. Asterisk can be used on its own for VoIP only functionality. Refer to **www.asterisk.org** for information and support. Asterisk is a registered trademark of Digium. Asterisk is started automatically at system startup. It runs under the control of the "asterisk" user (as opposed to the "root" user). Asterisk is controlled and monitored by **Linux Service Management (pg. 21)**. Using the "stop" commands at the Asterisk console will not stop Asterisk as the service management software will automatically restart it. To stop Asterisk, execute the following at the Linux prompt on the appliance: svc -d /service/asterisk To start Asterisk, execute the following at the Linux prompt on the appliance: svc -u /service/asterisk Note that executing any of the "restart" commands from the Asterisk console will still restart Asterisk as expected. Refer to **Making Asterisk Calls** in the Appliance User Guide for a list of default extensions supplied to use the appliance hardware. Section **Advanced Configuration** in the Appliance User Guide provides appliance specific information about Asterisk configuration. If Asterisk is configured to send voicemail files to users via email, a mail server to forward the emails is required. Refer to **Network Applications (pg. 29)** for a list of available mail server packages.

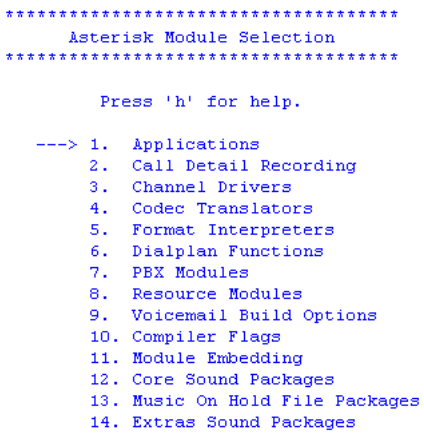
Zaptel version 1.4.9.2	zaptel	none	While the appliance provides the hardware clock, Zaptel provides clocking support specifically for Asterisk. While most features of Asterisk will function without Zaptel, Zaptel clocking improves the performance of Asterisk when used with PIKA hardware. Asterisk MeetMe conferencing will not function on the appliance without Zaptel clocking. Refer to **www.asterisk.org** for information and support.
PIKA channel driver version 3.6.x	chan_pika	HMP, Zaptel	This package enables Asterisk to use the appliance hardware (FXO, FXS and BRI ports, audio line in and line out ports). The channel driver augments the clocking supported provided by Zaptel. The channel driver utilizes the HMP package to control the appliance hardware. Configuration settings for the channel driver are located in the file /persistent/etc/asterisk/pika.conf. Documentation for the configuration file is located at **http://www.pikatechnologies.com/english/View.asp?mp=463&x=605.** Select the link for ***chan_pika.html***.

Astmanproxy	astmanproxy	Asterisk, LCD library	This package installs the astmanproxy service which uses the Asterisk Management Interface (AMI) to display Asterisk call status information on the appliance LCD. The service is started at system startup. If Asterisk is stopped for any reason, the LCD will display "Attempting to Connect ..." as the astmanproxy service attempts to connect with Asterisk. After 10 retry attempts, if the connection cannot be established, the message "Connection Failed" will appear on the display. If a phone or trunk is in use before the astmanproxy service starts, the status on the display for that line will not be reflected until the next time the device is used. To run the astmanproxy service in debug mode, execute the following commands at the Linux prompt on the appliance: killall astmanproxy astmanproxy -d Logs will be generated to /persistent/var/log/asterisk/astmanproxy.log. Use of debug mode should be limited, as the logs will consume space on the persistent file system.
Asterisk GUI	asterisk-gui	Asterisk	The Asterisk GUI provides a web-based graphical interface to configure Asterisk and to upgrade the appliance software. The GUI uses web server functionality built into Asterisk and does not require a separate web server package. Refer to **Asterisk Configuration Using the Asterisk GUI** in the PIKA WARP the Appliance User Guide for information about the Asterisk GUI screens that are specific to the appliance.

Customizing Asterisk

A set of default Asterisk options has been preselected. To use different Asterisk features, select the Asterisk Custom Settings option under the Asterisk menu. When Asterisk is compiled, the following menu

will be presented:



When the required features have been selected, press 's' to save changes or 'q' to quit. Compilation will continue.

7.4 Timezone

This package provides a utility that allows you to specify information about your time zone so that the date and time used on the appliance reflect local time instead of Universal Time (UTC). The package directory name is zoneinfo and it has no dependencies.

Time zone information for both the Americas and Europe is included in the software image shipped with the appliance. A separate menu option to include full time zone information for all countries is also provided. To use the utility, enter the following command at the Linux prompt on the appliance:

```
timezone
```

You will be presented with a set of menus to select your time zone. If your city does not appear in the list, chose the one closest to you. Changes will take affect immediately for the appliance itself. Asterisk must be restarted to use the new time zone setting.

If you do not wish to install the "Timezone Info" package, time zone information can be set by including the following line in the files /persistent/autorun/S32ntpd and in /persistent/etc/localenv.

```
export TZ=<time zone code>
```

Appendix B (pg. 115) lists time zone codes for most countries.

You will need to reboot your system for the changes to take effect. Note that Asterisk will use UTC unless the "Timezone Info" package is used.

7.5 Applications

The following packages are located in the Applications menu. These packages are not included in the factory default appliance software.

Package Name	Package Directory	Dependencies	Description
SQLite	sqlite	none	SQLite is an open source software library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. For support and further information about SQLite, refer to **http://www.sqlite.org**.
PHP	php	sqlite	PHP is a general purpose scripting language intended for use in web-based applications. For support and information, refer to **http://php.net**.

7.6 Network Applications

The following packages can be selected from the Networking menu.

Package	Directory Name	Present in Default Software	Description
SSH Client and Server	dropbear	yes	Dropbear is an open source implementation of the SSH and SCP protocols and provides remote shell access to the appliance. It provides the client, server, key generation, and key encryption. It will generate the necessary encryption keys the first time it is started after replacing the persistent file system image. Most users will want to include this package for development purposes. For information and support, refer to **http://matt.ucc.asn.au/dropbear/dropbear.html** Note that to use SSH from the appliance to access another system, the client executable is called dbclient. Dropbear is controlled by **Linux Service Management (pg. 21)**.
TFTP Server	tftpd	yes	Tiny File Transport (TFTP) is used as a bare-bones special purpose file transfer protocol. TFTP allows only unidirectional transfer of files, depends on UDP, has low overhead, and provides virtually no control. TFTP provides no user authentication. TFTP is a high security risk and should not be enabled unless absolutely necessary. The package adds only the server side of the protocol, a tftp client is included by default and is not controlled by selecting this package. The server directory is /persistent/tftpboot.
FTP Server	ftpd	no	File Transfer Protocol (FTP) server support is provided by the vsftp (Very Secure FTP) package, an open source implementation of a complete, session-oriented, general purpose file transfer protocol. Refer to **vsftpd.beasts.org** for details and support. FTP on the appliance is enabled for **read-only** anonymous access. Files in the directory /persistent/ftp on the appliance can be retrieved using any standard FTP client, such as Filezilla.

NTP Client and Server	ntpd	yes	Network Time Protocol (NTP) is a protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. NTP is configured to use the cluster of time servers supported by **http://www.pool.ntp.org**. Files in the package directory (<Your PADS Path>/packages/ntpd) are used to initialize and configure the NTP service. • The file ntp.conf contains configuration information for the service, including the list of servers used for time synchronization. This file is copied to /persistent/etc when the ntpd package is built. • The file "run" starts the NTP service. This file is copied to /service/ntpd when the ntpd package is built. • The file S32ntpd (copied to /persistent/autorun when ntpd is built) sets the date initially using ntpdate. This ensures that the time is correct before the NTP service starts. After system initialization, the ntpd service maintains the correct time using the servers specified in the file /persistent/etc/ntp.conf for synchronization. The ntpd service is controlled by **Linux Service Management (pg. 21)**.
HTTP Server	httpd	yes	This options selects the BusyBox web server and configures it. The default location for web pages is /persistent/var/www/htdocs.

Forwarding SMTP	smtp, nullmailer	no	Two mail server packages are available under the Forwarding SMTP sub-menu.
			• ssmtp - very simple, but obsolete
			  • configuration file: /etc/ssmtp
			• nullmailer - more powerful and actively supported
			  • configuration file: /etc/nullmailer/remotes
			To use a mail server, the appropriate configuration file must be updated with the proper settings for your mail setup.
			
			If Asterisk is configured to send voicemail files to users via email, a mail server to forward the emails is required.

7.7 ext2 File System Utilities for USB and SD Media

This package is located in the Utilities menu and the package directory name is e2fsprogs. It has no dependencies.

While the appliance supports different types of files systems, any file system used by Asterisk must have file locking capability. The file system supported on the appliance with this capability is the Linux second extended file system (ext2). This package provides utilities to assist in formatting and checking ext2 files systems which may be required to use the SD card or a USB key (which, by default, are typically formatted as FAT) with Asterisk. Two utilities are provided:

- e2fsck
  - Used to check a Linux second extended file system (ext2fs). Note that, in general, it is not safe to run e2fsck on mounted file systems. However, even if it is safe to do so, the results printed by e2fsck are not valid if the file system is mounted. If e2fsck asks whether or not you should check a file system which is mounted, the only correct answer is "no".
- mk2fs
  - Used to create an ext2 filesystem (usually in a disk partition).

Scripts format-sd and format-usb are provided to format the SD and the USB, respectively, to ext2 format. The following table shows the usage of the utilities and the scripts. Before using these utilities, ensure that the corresponding device is unmounted according to the instructions below. In all of the examples, the commands are executed at the Linux prompt **on the appliance**.

Commands	Purpose
umount /dev/mmcblk0p1 format-sd /dev/mmcblk0	Unmounts partition 1 on the SD and formats a single partition on an SD card to ext2.
umount /dev/sda1 format-usb /dev/sda	Unmounts partition 1 on the USB drive and formats a single partition on a USB drive to ext2
umount /dev/mmcblk0p1 e2fsck /dev/mmcblk0p1	Unmounts partition 1 on the SD and does a sanity check on the file system
umount /dev/sda1 e2fsck /dev/sda1	Unmounts partition 1 on the USB drive and does a sanity check on the file system

The USB drive is not automatically added to the Linux file system in the appliance, regardless of whether it is inserted into a system at run-time or before system startup. Once you have inserted the USB drive, to add it to the file system, enter the following command at the Linux prompt on the appliance:

```
mount /mnt/usb
```

Before removing the USB from a live system, you must unmount the USB key from the file system or data may be lost. Enter the following command at the Linux prompt on the appliance before removing the USB device. Note that if any applications have files open on the USB, the applications must be stopped first.

```
umount /mnt/usb
```

By default, the USB is mounted as a FAT file system format. If the USB is formatted for ext2, this entry in /etc/fstab must be changed:

```
/dev/sda1 /mnt/usb vfat noauto 0 0
```

to the following:

```
/dev/sda1 /mnt/usb ext2 noauto 0 0
```

More information about file systems and changing entries in /etc/fstab is provided in section **File System Layout (pg. 74)**.

For more information about the SD card and USB port, refer to **Using an SD Card** and **Using the USB Port** in the PIKA WARP the Appliance User Guide.

7.8 Software Update Utilities

The warploader utility provides the ability to write software images to flash memory while the appliance is running. It can be selected from the Utilities menu and the package directory name is warploader.

Refer to section **Writing Software Images to Flash Using the Warploader (pg. 69)** for usage information. The warploader is the preferred method for writing images to flash, therefore, this tool should be included in all development loads. It is also required to use the **Autoflash Feature (pg. 67)**.

7.9 Debugging Utilities

The GDB debugger is available to provide standard Linux debugging capabilities. It can be selected from the Utilities menu and the package directory name is gdb.

By default, all executables in the ramdisk are stripped of symbols when the software is built to reduce the image size. To make symbols available for debugging, select the menu option "Do Not Strip Executables" to include unstripped binaries for debugging.

Including this package and unstripped binaries greatly increases the size of the image and should only be used during development. Ensure that your system is set up to boot from NFS when using this package.

Debugging Asterisk

To get a proper stack trace, you must modify the default Asterisk build settings. Ensure that you start with a fresh compile of Asterisk by doing a "make asterisk-clean".

- Select "Asterisk Custom Settings" from the PADS main menu.
- During the compile, at the beginning of the Asterisk compile, you will be presented with a menu to configure Asterisk.
  - Select 10, Compiler Flags and check DONT_OPTIMIZE and optionally, MALLOC_DEBUG.
  - Press 's' to save the changes and compilation will continue.

7.10 Samples

The following sample applications can be selected from the Samples menu. No samples are included in the factory default software load.

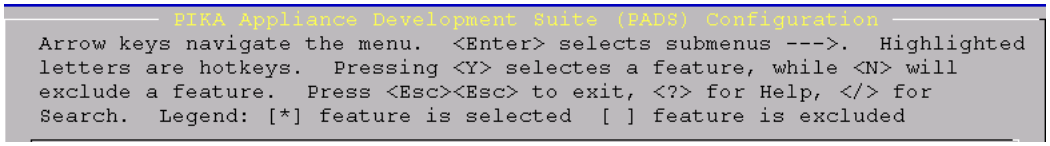
Package	Directory Name	Dependencies	Description
hello world	hello_world	none	This is a simple "hello world" application included to assist developers in adding packages to PADS.

IVR Application	monzaivr	HMP, LCD library	This application demonstrates the use of the PIKA HMP SDK to create a telephony application. It can receive incoming calls from a SIP IP phone configured for direct IP calling. Upon connection, an announcement is played, then a tone is played, after which the user can begin recording. This sample includes code to update the appliance LCD with call status information. **This package cannot co-exist with Asterisk as there will be a conflict in the use of the SIP port.**

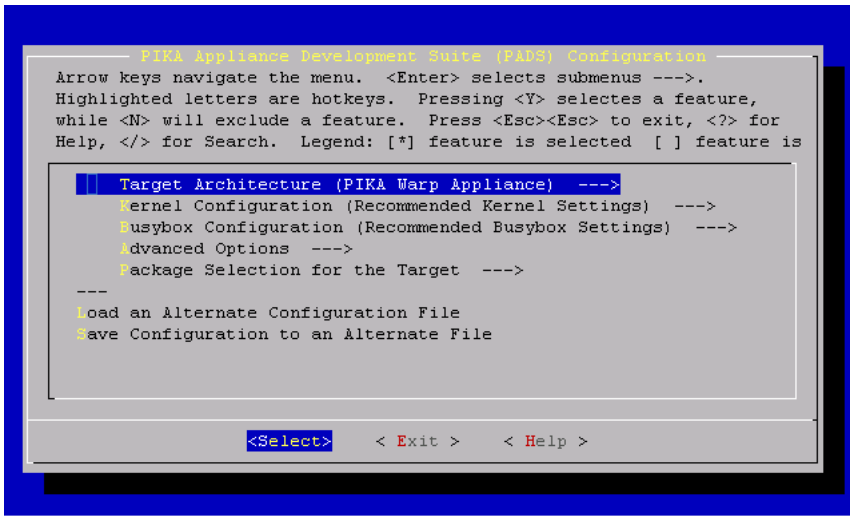
8 Navigating the PADS Menu

The PADS menu is displayed by running 'make menuconfig' from the top level directory of PADS (<Your PADS path>).

Package selection in PADS is driven by an ncurses-based menu system. The information at the top of the menu describes how to navigate the menu and select components.



The following is the main menu:



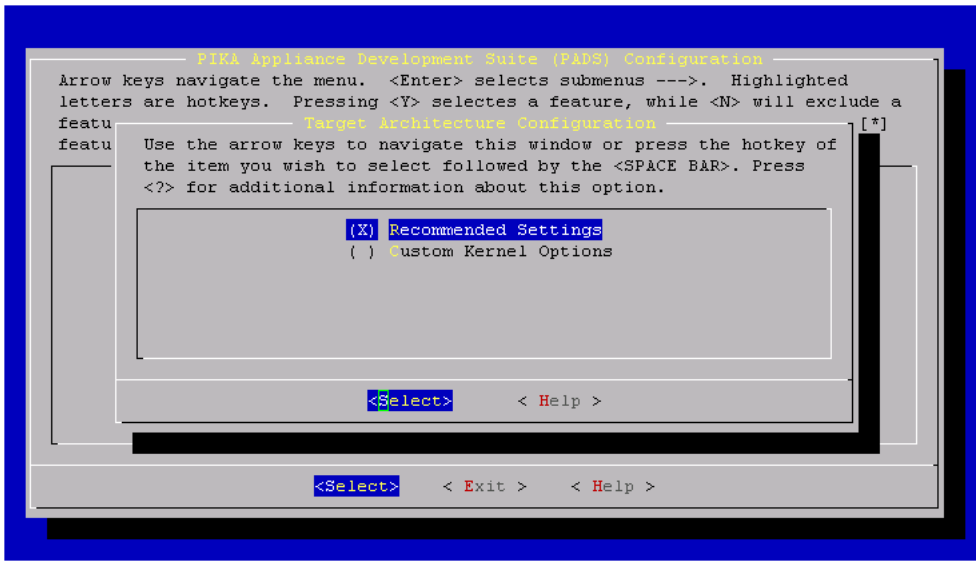
8

The menu item "Target Architecture (PIKA Warp Appliance)" indicates that the currently selected architecture is PIKA WARP the Appliance. At this time, there is only one Target Architecture supported in PADS. The next sections describe the menus:

- Kernel Configuration
- Busybox Configuration
- Advanced Options
- Package Selection for the Target

8.1 Kernel Configuration Menu

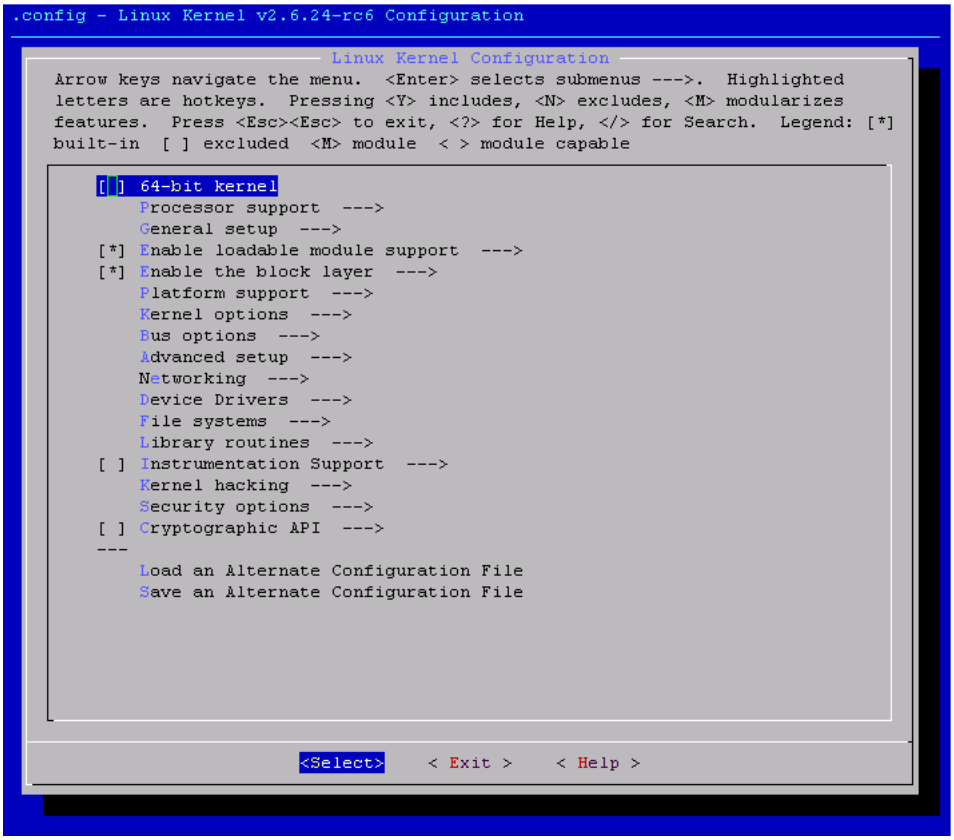
The operating system (kernel) software has been customized for the appliance architecture. This menu option provides the ability to further customize the kernel features used with the appliance. To use additional operating system features, select "Kernel Configuration" and you will be presented with the following menu:



"Recommended Kernel Settings" is the default option. Most users should stay with the default kernel settings. For specialized applications, alternative kernel settings may be required, in which case "Custom Kernel Settings" may be selected.

If "Custom Kernel Options" is selected, the following menu will be shown **when the kernel component is compiled** after entering "make" to begin compiling the software. Navigate the menu and select the required kernel options. After exiting the top-level kernel configuration menu, kernel compilation will continue.

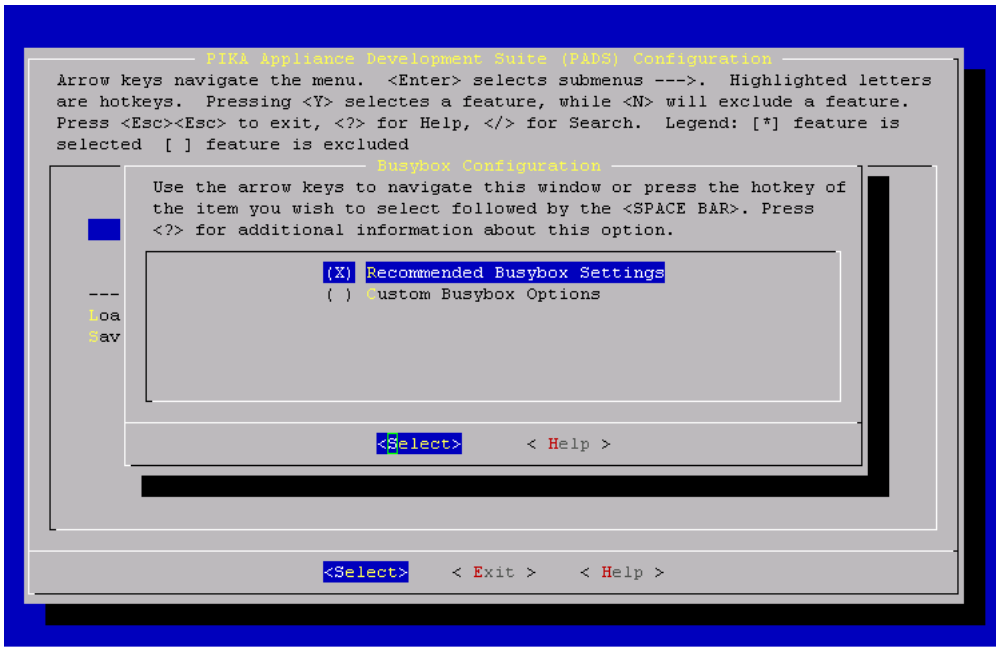
NOTE:	If you have previously compiled the kernel and then select "Custom Kernel Options", you will need to execute "make linux-clean" before entering "make" to begin compilation.



8.2 Busybox Configuration Menu

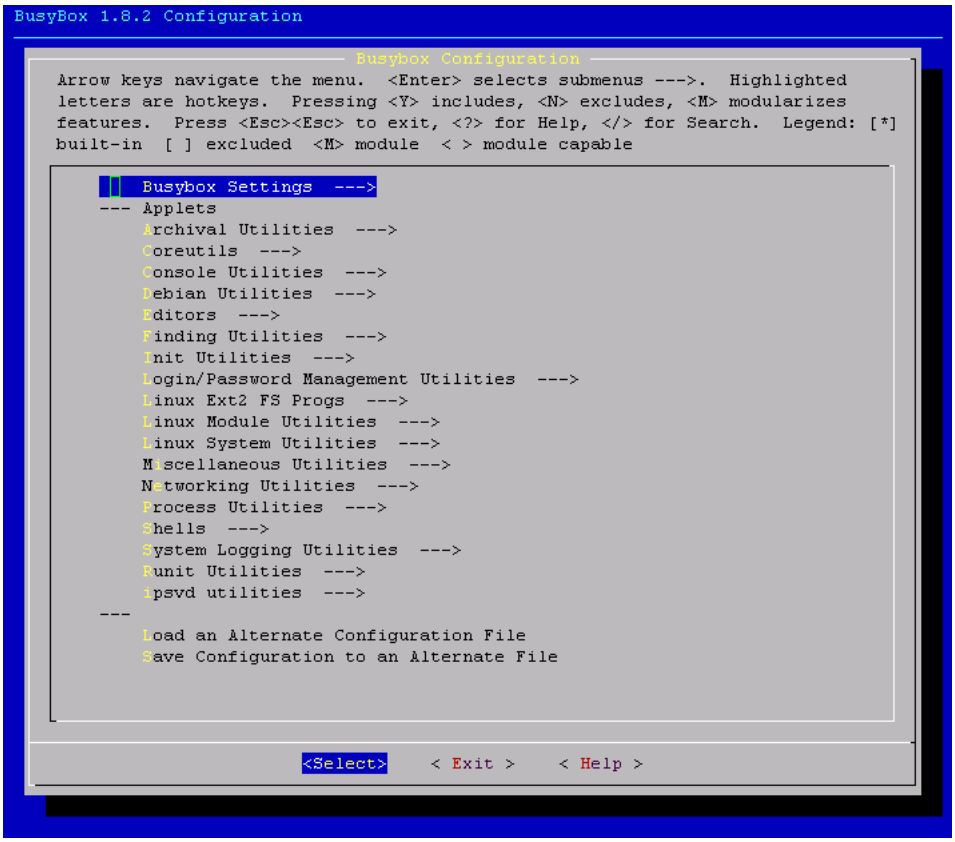
Refer to section **BusyBox (pg. 20)** for details about the this package.

A preselected set of features has been chosen from the Busybox package to use on the appliance. If different features are required, Busybox Custom options may be selected. When the Busybox Configuration Menu is selected, the following menu will be presented:



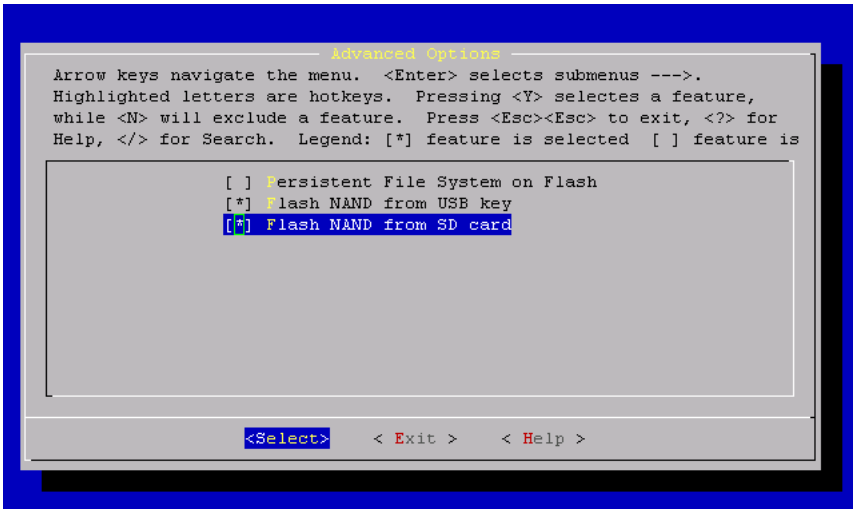
If custom options are selected, the following menu will be shown **when the BusyBox component is compiled** after entering "make" to begin compiling the software. Select any additional tools you would like included in the image. Note that the size of the image will increase in proportion to the number of additional tools selected. Refer to **Managing the Ramdisk Image Size (pg. 49)** for information about the image size.

NOTE:	If you have previously compiled busybox and then select "Custom Busybox Options", you will need to execute "make busybox-clean" before entering "make" to begin compilation.



8.3 Advanced Options Menu

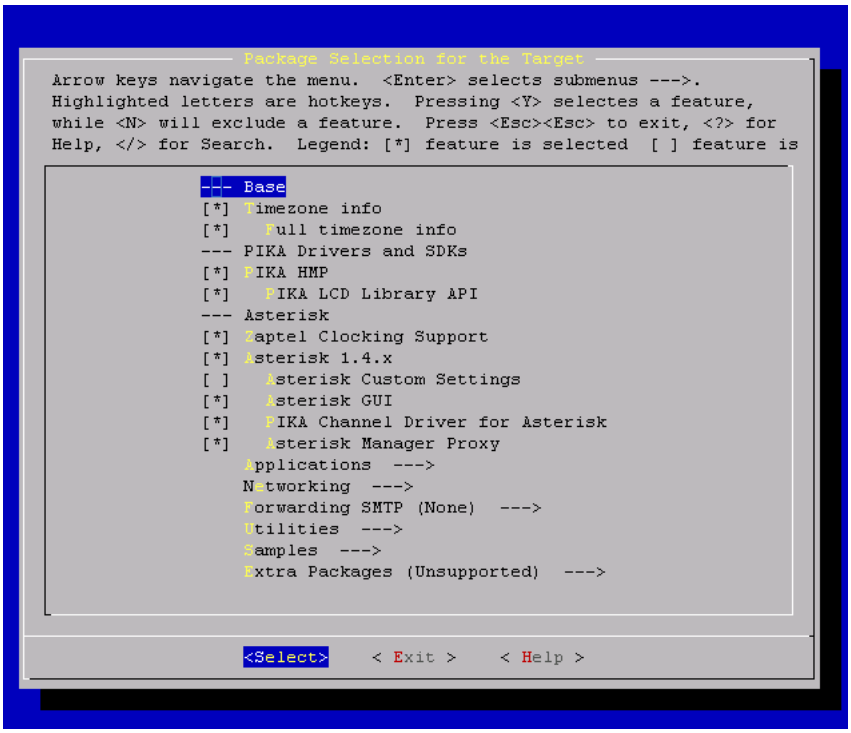
The following shows the advanced options menu.



The options presented in this menu are intended for advanced users. Refer to **Using the Persistent File Systems from Flash (pg. 83)** and **Using the Autoflash Feature (pg. 67)** for information about using these options.

8.4 Package Selection Menu

When the menu option "Package Selection for the Target" is selected from the top-level menu, the following menu is displayed. Each of the packages under this menu and the sub-menus is described in **Software Packages Available in PADS (pg. 18)**. The "--->" indicates a sub-menu, press the enter key when the item is selected to enter the underlying menu.



8.4.1 Extra Packages

Extra Packages provides menu access to all packages in **http://svn.pikatech.com/pads/extra_packages**. Packages are automatically retrieved and displayed in the menu. Any selected package will be included at compile time.

NOTE:	**Extra packages are not supported by PIKA. They are provided "as is" to help customers use these packages in a cross-compile environment.**

9 Developing Software for the Appliance

This section provides information about developing your application for an embedded system.

- **Design Guidelines for an Embedded System (pg. 43)** - Describes items to consider when writing an application for an embedded system.
- **Using the Additional Persistent Flash Memory (pg. 44)** - Describes how to use the optional sections of flash memory.
- **System Initialization (pg. 45)** - Describes system initialization and how to include initialization steps for your application.
- **Managing the Ramdisk Image Size (pg. 49)** - Describes how to handle larger ramdisks.

9.1 Design Guidelines for an Embedded System

User applications should be designed to take into account the limitations and considerations of embedded systems compared to a standard PC. This section provides some guidelines for designing user applications for the appliance.

The appliance is intended for dedicated telephony applications and the available resources are sufficient for this use. However, unlike a regular PC, which is intended for a wide variety of applications, the appliance has limited resources to run many applications simultaneously. Applications that consume significant CPU and memory can compete for resources, causing degraded voice quality.

When designing your application, you should consider:

- **Memory management** - The entire file system (ramdisk) is loaded into RAM at run-time and consumes a predefined amount of memory. Applications that use a significant amount of memory may experience reduced performance due to memory paging. The maximum size of the ramdisk, based on the current settings, is 123 Megabytes, that is, it will consume 123M out of 256M of RAM. This size is sufficient for a load that includes all of the packages currently made available by PIKA in PADS (not including extra packages), with the exception of GDB. Some techniques to reduce memory usage include the following:
  - Use persistent storage for files that are typically read-only. Refer to **Using the Additional Persistent Flash Memory (pg. 44)** for alternative locations for file storage. Avoid the use of persistent storage for files that require write access during run-time and whose contents need not be preserved across reboots. Writing to flash is slower than writing to RAM and frequent, unnecessary writes may shorten the life of the flash memory. It may also impact voice quality for voice

applications.

- Consider alternative approaches for files that are written frequently at run time and which may consume a lot of RAM, such as logging and voice mails. If files such as these are stored in RAM, not only will the information be lost after a reboot, but RAM may be consumed unnecessarily. It is recommended that these types of files are directed to alternate storage such as the SD or USB, or in the case of logging, to an off-board system. Refer to **Logging (pg. 76)** for more information. For the reasons stated in the previous bullet, writing these files to persistent flash memory is not recommended.
- Consider the use of temporary file systems for files that are frequently updated and need not be preserved across reboots. Entries in the file /etc/fstab define the directories that are designated as temporary files systems and specify the maximum amount of RAM they may consume. Space in RAM is only used when files are written to these directories (space is not set aside up front). For example, the directory /var/log is a temporary file system which is limited to 1 megabyte of RAM. Refer to **File System Layout (pg. 74)** for more information.

- **Byte order** - The PowerPC processor is big-endian (most significant byte is at the lowest address), while the typical PC, which uses an x86 processor, is little-endian (least significant byte is at the lowest address). If you are porting your application from another system, you show consider the following items. Applications written only for the appliance typically will not have issues.
  - The headers for audio files created on a little-endian system will be incompatible. For example, any recordings, such as IVR prompts, created on your x86-based development computer will require appropriate code in your application to interpret the headers correctly when the files are played from within your application.
  - Any application that communicates data across a network with other applications should respect network byte ordering for messaging.

9.2 Using the Additional Persistent Flash Memory

Two additional partitions are provided in flash memory for user-defined purposes. This space can be used for as part of a memory management strategy or for data that must survive system reboots. The partitions are shown in the **NAND Flash (pg. 65)** section as persistent 1 and persistent 2.

In your application .mk file in the main package target section, copy the appropriate files into either persistent 1 or persistent 2 using the following makefile variables:

$(PERSISTENT1_STORAGE) - persistent1 flash memory partition
$(PERSISTENT2_STORAGE) - persistent2 flash memory partition

"make image" will automatically create images for the additional partitions **if there are files present in the associated directories**.

- persistent1 - image name image1.jffs2
- persistent2 - image name image2.jffs2

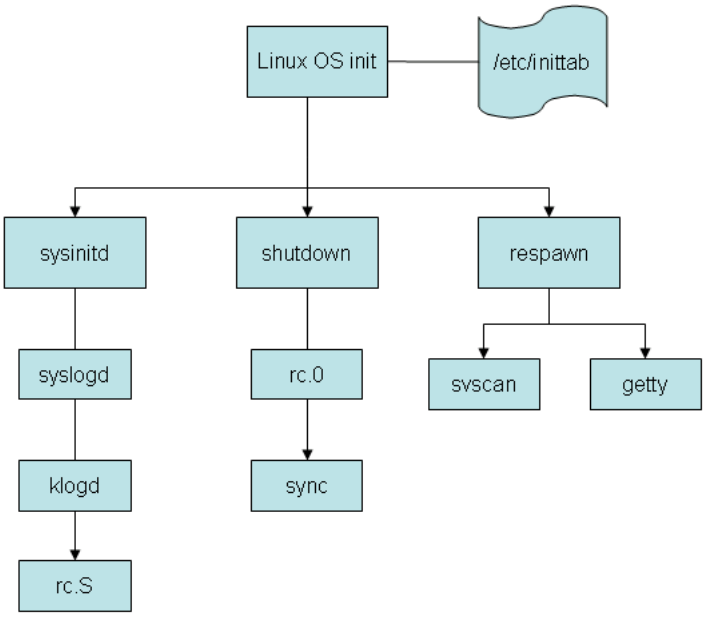
Both partitions will be mounted at run-time as /persistent1 and /persistent2.

If the option "Persistent File System on Flash" is **not** selected (refer to **Advanced Options Menu (pg. 40)** for more information), in additional to the regular persistent file system, both persistent1 and persistent2 will be located on your development computer. The additional persistent file systems will be included in the NFS mount point at <Your PADS Path>/build_warp/root/persistent1 and <Your PADS Path>/build_warp/root/persistent2. Files changed in these directories will immediately be reflected on the appliance.

9.3 System Initialization

User applications may need to perform certain actions during system startup. This sections describes the initialization sequence and how to include application initialization steps into the process.

The file /etc/inittab contains the set of actions that the appliance follows for system initialization, shutdown and actions for normal operations.



At present, the only actions for normal operation are those that spawn processes:

- getty - listens for input on the serial port
- svscan - service monitoring process (refer to **Linux Service Management (pg. 21)**)

The script rc.0 (included with the skeleton package) contains the steps to execute upon system shutdown. To avoid data loss, the script ensures that any data buffered in RAM is written to permanent storage (SD, USB, flash) and unmounts the SD. This script will not execute on a hard reset and therefore, it is advisable to perform a software reset when possible.

Initialization for the appliance is controlled by the rc.S script(included with the skeleton package). The following table describes the main actions in rc.S.

Action	Description
Mount the base file systems, including the persistent file system	Only the base file system can be mounted from this script, the file /persistent/etc/fstab defines the other file systems.
Create softlinks between files in /persistent/etc and /etc	Persistent configuration files are linked into /etc/ for normal system operation.
set HOSTNAME	Refer to **Network Settings (pg. 79)** for information about when this value is used.
mount all partitions except those marked noauto in fstab	Mounting options for partitions in fstab may specify: • auto - mount automatically at system startup • noauto - do not mount automatically at system startup, it can be mounted manually later • size - mount the file system automatically with the size specified, limits the amount of RAM that can be used for this file system Refer to **File System Layout (pg. 74)** for more information.
mount the SD	SD is only mounted if there is a card present
execute scripts in /persistent/autorun in ascending numerical order	Autorun scripts contain initialization instructions for each package in the system. Scripts in the /persistent/autorun directory have the format 'S', a number, and the package name. The scripts are executed in numerical order, starting with the lowest number. The numbers are used strictly for determining the order in which the files will be executed. A separate file must be created for each package that requires initialization.
start the watchdog	Software watchdog, timeout 15 seconds

execute commands in rc.local	User-defined initialization steps which are specified in rc.local (located in the skeleton package). In most cases, it should not be necessary to modify the contents of the rc.S script. Any system-level application specific steps should be added to rc.local. Initialization for individual packages should use an autorun script.

Options for executing your initialization steps are:

1. Use an autorun script. This is useful if your package depends on the initialization steps of another package. To use an autorun script, perform the following steps:
   - Create a file with the initialization steps.
     - The number you use in the file name must reflect the priority of your package initialization in relation to other packages; the number should be higher than that of the packages that must initialize before yours. Multiple packages can use the same number.
     - Ensure that the initialization script file is executable.
     - The script should include a command indicating service it is starting.
   - Add the script to the directory /persistent/autorun directory as one of the steps in the main target rule in your package .mk file.
2. If you would like your service to be controlled using dameontools (refer to **Linux Service Management (pg. 21)** for details), a "run" script can contain initialization instructions.
   - Create a file called "run" which contains the commands to initialize and start your service.
   - Ensure that the run file has executable permissions.
   - Add a line in the <package>.mk file to copy the run file to $(TARGET_DIR)/service/<your service name>/run.
   - If you would like to log service information, perform the following steps:
     - Create a file called "log-run" which uses the 'multilog' command to specify where you would like the logs to be stored (e.g. /var/log/<your service name>)and include it under your package directory.
     - Add a line in the <package>.mk file to copy the log-run file to $(TARGET_DIR)/service/<your service name>/log/run.
3. For system level initialization, add instructions to rc.local.
4. Modify rc.S if there are system level actions to perform that must occur between steps that are currently executed in rc.S. For example, if there is an action to execute that must occur before the autorun scripts are executed, it would be necessary to modify rc.S.

Autorun or run scripts typically execute the command to start a service, in addition to any other initialization steps. An application may use both an autorun and a run script, if necessary. Note that the run scripts are executed after rc.S completes, and therefore, will occur after the execution of all autorun scripts. Run scripts are executed alphabetically be service name.

Example

The httpd package uses the file S51httpd to perform actions at initialization. The number 51 in the file name indicates that it has medium priority in relation to other packages. For example, the script that sets up networking must run before the script for httpd. The networking initialization script has priority 30, so using 51 for httpd ensures that it will run after networking initialization.

The httpd initialization script is shown below. It performs the following actions:

- Sets up a softlink between the directory in persistent storage where web pages are stored to the location in the ramdisk where a web server will access them at run time.
- Starts the web server service which includes specifying the location of the configuration file.

```
#!/bin/sh

[ -d /var/www ] || ln -s /persistent/var/www /var/www

echo "Starting HTTPD"
httpd -c /etc/httpd.conf -h /var/www/htdocs
```

NTP uses both an autorun and a run script. The autorun script, S32ntpd, shows that it executes a command that must run before the ntpd service starts.

```
#!/bin/sh

# The ntpd service is started and monitored by the daemontools.
# The action in this script sets the date initially because we
# want the time to be correct before starting daemontools or
# svstat will report bogus times. After system initialization,
# it is up to the ntpd service to maintain the correct
# time. The file /etc/ntp.conf specifies the servers that
# the ntpd service will use for syncronization.
#
# For information about the daemontools and ntp, refer to the PADS User Guide.
#
# ntpd -q takes too long, so we use ntpdate instead.
ntpdate 0.pool.ntp.org
```

The run script for ntpd starts the ntpd service.

```
#!/bin/sh
```

```
logger "Running ntpd..."
exec /usr/sbin/ntpd -g -n 2>&1
```

9.4 Managing the Ramdisk Image Size

As described previously, memory usage is an important consideration for system performance. It is also important to determine whether the final ramdisk image will fit into the space allocated in RAM. Depending on how you have organized your software, you may find that there are issues with the size of the ramdisk when the image is built. This section describes options to handle this situation.

If you add new packages and the following error appears while building the images, you have exceeded the currently allocated space.

```
bin/genext2fs: couldn't allocate a block (no free space)
make: *** [image] Error 1
```

If this occurs, there are two options available:

1. Move some files to the additional persistent flash memory (refer to **Using the Additional Persistent Flash Memory (pg. 44)**)
2. Increase the size of the image (not recommended)

Increasing the size of the image is not recommended as the current size uses nearly half of the available RAM, regardless of the actual image size. However, should you choose to do so, you will need to make the following changes.

1. Increase the value of ROOTFS_SIZE in the top level Makefile of <Your PADS Path>
   - The size is specified in blocks, where a block is 1024 bytes.
   - To determine the size required, in the directory <Your PADS Path>/build_warp/root, execute the command "du -sk --exclude=persistent*". This will return the size of your ramdisk in blocks, excluding the persistent file systems, which will not be included in the image.
2. In U-Boot, increase the value of **ramdisk_size** in the environment variable ***ramargs***
   - **ramargs**=setenv bootargs root=/dev/ram rw **ramdisk_size=130000**
   - This value must be the same as ROOTFS_SIZE in the Makefile

Note that the numbers above reflect the **uncompressed** size of the ramdisk. The command "make image" creates a compressed image. The actual size of the image can vary based on the compression rate when the image is created, which depends on the type of files in the image. At boot time, U-Boot decompresses the ramdisk image in flash and loads it into RAM. The U-Boot environment variable

load_nand_ramdisk specifies the address in RAM to start writing, the address in NAND to start reading and the number of bytes (in hexadecimal) to write from NAND into RAM.

```
load_nand_ramdisk nand read.jffs2 2200000 200000 3000000
```

The number of bytes to read is based on the **compressed** size of the ramdisk image. The current value specified in the environment variable is sufficient for a ramdisk image size of 48 Megabytes. When the image is built, if it is too large to boot successfully, a warning is generated:

```
WARNING!!!! Ramdisk image is too large and will not boot.
Size must be less than 48M
```

If you attempt to boot the image, the following will be displayed at the serial console at boot time:

```
## Loading RAMDisk Image at 02200000 ...
Image Name: PIKA Warp 2.0.0-60
Image Type: PowerPC Linux RAMDisk Image (gzip compressed)
Data Size: 44368879 Bytes = 49.5 MB
Load Address: 00000000
Entry Point: 00000000
Verifying Checksum ... Bad Data CRC
```

Should it be necessary, you can increase the last number in the U-Boot environment variable ***load_nand_ramdisk*** to a value sufficient for your image size. You should avoid increasing the image size arbitrarily as the time required to boot the image will increase accordingly.

10 Adding a Package to PADS

This section describes how to add your own applications to PADS and include them in the menu system. It also provides more information about using PADS.

To add a package to PADS, take the following steps:

1. Add a directory under <Your PADS path>/package with the name of your package.
2. In the new directory, add two new files, <package>.mk and Config.in.
3. Fill in the files <package>.mk and Config.in with information appropriate for your package. This will include settings for cross-compiling.
4. Add your new package to the file <Your PADS path>/package/Config.in.

The following sections describe these steps in more detail.

In the top level PADS directory <Your PADS path>, there are three important directories:

```
<Your PADS path>/
        build_warp/
        dl/
        package/
```

Directory Name	Purpose
package	Each package has its own directory containing the files specific to the package. Two files must be present for each package: • <package>.mk - makefile that defines how a package is built in PADS • Config.in - specifies information about adding the package to the Package Selection menu Details about these files will be explained in subsequent sections. Additional files for each package may include a patch file, an autorun directory that contains a file specifying initialization instructions and any other necessary configuration files. Further information about initialization instructions can be found under **System Initialization (pg. 45)**.

dl	Package source files are placed in this directory when they are initially obtained at compile time, either as tarballs downloaded from an FTP site or checked out from a source code repository. Currently, the only repository supported by PADS is Subversion (SVN).
build_warp	This is the top level directory used for compiling packages for the appliance architecture. Each package has its own subdirectory which is created when it is unpacked from the **dl** directory. The directory structure created at compile time is used when images for the ramdisk or persistent file system are built. The directory build_warp/root/persistent is the basis for the persistent file system image. **The directory build_warp/root is the directory on which the ramdisk image is based and also can be used as an NFS mount point.**

Global Variables

A number of global variables are used in the .mk files. They can be found in <Your PADS path>/package/Makefile.in. The following table describes the variables and their usage.

Variable	Value	Description
BASE_DIR	<Your PADS Path>	Directory where you downloaded or checked out PADS code
DL_DIR	$(BASE_DIR)/dl	Download directory for package source
BUILD_DIR	$(BASE_DIR)/build_warp	Directory used for compiling packages
TARGET_DIR	$(BUILD_DIR)/root	The root file system, basis for the ramdisk image and the NFS mount point
PERSISTENT_STORAGE	$(BUILD_DIR)/root/persistent	Directory for the persistent file system, basis for the persistent file system image **NOTE**: The value for this variable can change depending on whether the option "Persistent File System on Flash" is selected in the **Advanced Options Menu (pg. 40)**. Refer to **Using the Persistent File Systems from Flash (pg. 83)** for details about this option.

PERSISTENT_STORAGE1	$(BUILD_DIR)/root/persistent1	Directory for the first additional persistent storage area **NOTE**: The value for this variable can change depending on whether the option "Persistent File System on Flash" is selected in the **Advanced Options Menu (pg. 40)**. Refer to **Using the Persistent File Systems from Flash (pg. 83)** for details about this option.
PERSISTENT_STORAGE2	$(BUILD_DIR)/root/persistent2	Directory for the second additional persistent storage area **NOTE**: The value for this variable can change depending on whether the option "Persistent File System on Flash" is selected in the **Advanced Options Menu (pg. 40)**. Refer to **Using the Persistent File Systems from Flash (pg. 83)** for details about this option.
TARGETS	Depends on the packages selected	List of targets to be built, based on packages selected using the menu

10.1 The Package .mk File

The .mk file defines how a package is downloaded, configured, compiled, and installed. This section assumes you are familiar with makefile concepts.

10.1.1 Variables

In the package .mk file, the following variables are recommended for each package:

Variable	Purpose
<package name>_VER	A version so that you can distinguish between different releases of the package.
<package name>_SOURCE	A distinct name for the source that will be obtained. It should include the version variable to ensure uniqueness.
<package name>_DIR	A distinct directory name used for compiling the package. It should include the version variable to ensure uniqueness. This is the subdirectory into which the source will be unpacked under <Your PADS path>/build_warp.
<package name>_SITE	The location of the package source. The location can be an FTP site or an SVN repository.

Other variables can be defined as required to improve the readability of the .mk file.

Example

This example is taken from the file dropbear.mk. Note the use of the version number as part of the package source name and the package build directory. The source is obtained from PIKA's FTP site.

```
ifeq ($(strip $(PADS_WARP)),y)
SSH_VER=0.50
SSH_DIR=$(BUILD_DIR)/dropbear-$(SSH_VER)
SSH_SOURCE=dropbear-$(SSH_VER).tar.gz
SSH_SITE=ftp://ftp.pikatech.com/outgoing/pads
SSH_UNZIP=zcat
endif
```

This example is taken from chan_pika.mk and shows an example of variable definitions for a package that obtains the source from an SVN repository. Note that the variable CHAN_PIKA_VER is actually a path within the SVN repository and it is used in the definition of the directory name used for compiling the package.

```
CHAN_PIKA_SITE=http://svn.pikatech.com
CHAN_PIKA_REPOSITORY = chan_pika
CHAN_PIKA_VER       = tags/1.1.0.62
CHAN_PIKA_SVN_REV=head
CHAN_PIKA_DIR_VER=$(shell echo $(CHAN_PIKA_VER) | sed s:/:-:g)
CHAN_PIKA_DIR=$(BUILD_DIR)/chan_pika-$(CHAN_PIKA_DIR_VER)
CHAN_PIKA_SOURCE=chan_pika-$(CHAN_PIKA_DIR_VER)
```

10.1.2 Rules

A set of rules must be defined in the .mk file for the package. Each rule has a prerequisite that determines the order of rule execution. The following rules may be defined:

Rule	Purpose
package source	Defines a target that obtains the package source from the remote site to the download directory
.unpacked	Defines a target and associated rules that decompress the downloaded package source. Depends on the package source rule. This may include applying patches to the package source. Patches are typically used to substitute platform-specifics in code written for a different architecture. For example, Makefiles written for an x86 platform may require different options for cross-compiling on the Power PC architecture of the appliance and it may be undesirable to permanently alter the original makefile. Patches should only be applied as part of unpacking as multiple attempts to apply a patch will fail.
.configured	Defines a target and associated rules that configure the software. Depends on the .unpacked rule. If a package has a configure script, it should be executed here.
main package target	Defines how to cross-compile and install the package. Typically uses the compile and install rules in the package source Makefile.
clean	Defines a target to clean the software build by calling the clean and/or uninstall rules from the package Makefiles. The clean target should run make clean on $(BUILD_DIR)/package-version and MUST uninstall all files of the package from $(TARGET_DIR).
dirclean	Defines a target to completely remove the directory into which the software was uncompressed, configured and compiled. The dirclean target MUST completely remove the package source directory from the $(BUILD_DIR).

The .configured and main package target rules will typically require settings to cross-compile your application. Different versions of the compiler and linker are required in order to cross compile for the PowerPC, as a development computer will only have versions suitable for an x86 processor. The

toolchain included with PADS provides versions of the compiler and linker that are required to build applications for the PowerPC.

Cross-Compile Settings in the .configured Rule

If your package has a configure script, there are typically options that can be passed into the script to define your environment. Typical variables include the following:

- --host=powerpc-linux
- --target=powerpc-linux
- --prefix=$(TARGET_DIR)/usr
  - Most applications and libraries install into /usr/local by default, however, this directory is not present on the appliance, and the libraries are located in $(TARGET_DIR)/usr
- --with-libraryX=$(TARGET_DIR)/usr
  - If your application requires a external library, the location on the appliance should be specified, otherwise, it will be linked against the version on your development computer.

Cross-Compile Settings in the Main Package Rule

The PADS environment uses the following variables to specify the location of compiler and linker. Depending on whether the package uses a configure script, these variables may be required when executing "make" from within the .mk file. The ARCH and CROSS_COMPILE variables may also be specified. These may only be necessary if the makefile performs different steps based on the architecture.

Variable	Description
ARCH=powerpc	Indicates the processor architecture
CROSS_COMPILE=ppc_4xxFP-	Cross-compile architecture to use from the toolchain
TARGET_CROSS=$(BASE_DIR)/toolchain/usr/bin/powerpc-linux-	Path to the cross compile tools
TARGET_CC=$(TARGET_CROSS)gcc	Location of the C/C++ compiler
TARGET_CXX=$(TARGET_CROSS)g++	Location of the C++ compiler. Typically only TARGET_CC or TARGET_CXX would be used.
TARGET_AR=$(TARGET_CROSS)ar	Location of the ar archive tool

Example

This example is taken from the dropbear.mk file. Rules are indicated in bold.

- The rule $(DL_DIR)/$(SSH_SOURCE) obtains the file from the FTP site specified using the variables

defined previously in the .mk file.

- The file is a tarball, so it is unpacked as part of the rule $(SSH_DIR)/.unpacked.
- The dropbear package source has a configure script which is called in the rule $(SSH_DIR)/.configured. The SSH_CONFIGURE_OPTS variable defines the options to pass to the script to indicate the environment.
- The main rule, dropbear, does the following:
  - Compiles and installs the the software using the compile and install rules defined in the Makefile with the dropbear source. It specifies the ARCH and location of the compiler as options when calling "make".
  - Copies configuration files into the persistent storage
- The clean uses the clean rule defined in the dropbear source Makefile. It also removes any files that were installed as part of the dropbear rule.
- The dirclean rule removes the dropbear build directory.

```
#Additional Variables
SSH_CONFIGURE_OPTS=--host=powerpc-linux --target=powerpc-linux  HOSTCC=gcc
CC=ppc_4xxFP-gcc ARCH=$(ARCH) --prefix=$(TARGET_DIR)/usr
SSH_ETC=$(PERSISTENT_STORAGE)/etc/dropbear

$(DL_DIR)/$(SSH_SOURCE):
   $(WGET) -P $(DL_DIR) $(SSH_SITE)/$(SSH_SOURCE)

$(SSH_DIR)/.unpacked: $(DL_DIR)/$(SSH_SOURCE)
   $(SSH_UNZIP) $(DL_DIR)/$(SSH_SOURCE) | tar -C $(BUILD_DIR) $(TAR_OPTIONS) -
   touch $(SSH_DIR)/.unpacked

$(SSH_DIR)/.configured: $(SSH_DIR)/.unpacked
   cd $(SSH_DIR); ./configure $(SSH_CONFIGURE_OPTS)
   touch $(SSH_DIR)/.configured


dropbear: $(SSH_DIR)/.configured
   $(MAKE) CC=$(TARGET_CC) BIN_DIR=$(TARGET_DIR)\
      ARCH="$(ARCH)"  \
      CROSS_COMPILE=$(CROSS_COMPILE) -C $(SSH_DIR) PROGRAMS="dropbear dbclient
dropbearkey dropbearconvert scp"

   $(MAKE) CC=$(TARGET_CC) BIN_DIR=$(TARGET_DIR) \
      ARCH="$(ARCH)"  \
      CROSS_COMPILE=$(CROSS_COMPILE) -C $(SSH_DIR) PROGRAMS="dropbear dbclient
dropbearkey dropbearconvert scp" install
```

```
    mkdir -p $(SSH_ETC)
    [ -f $(SSH_ETC)/dropbear_dss_host_key ] || \
        install -m 664 package/dropbear/dropbear_dss_host_key $(SSH_ETC)
    [ -f $(SSH_ETC)/dropbear_rsa_host_key ] || \
        install -m 664 package/dropbear/dropbear_rsa_host_key $(SSH_ETC)

    install -m 775 -D package/dropbear/run $(TARGET_DIR)/service/dropbear/run

    echo "Dropbear version" $(SSH_VER) >> $(PERSISTENT_STORAGE)/version_info.txt

dropbear-clean:
    if test -d $(SSH_DIR); then \
        $(MAKE) -C $(SSH_DIR) clean; \
    fi
    $(RM) -r $(SSH_ETC)
    $(RM) -r $(TARGET_DIR)/service/dropbear

dropbear-dirclean: dropbear-clean
    $(RM) -r $(SSH_DIR)
```

10.1.3 Compile Time Dependencies

Compile-time dependencies should be specified on the TARGETS line at the bottom of the .mk file. This line adds to the list of packages to compile and also specifies additional packages that must be compiled before the target. The .mk should first check if the package has been selected from the menu.

Specifying dependencies at the menu level (refer to **Adding Your Package to the Menu (pg. 59)** for more information) will ensure that the required dependencies are present but will not enforce compile/build order. If dependencies are not specified in this section, build order is alphabetical. Specifying the dependency here will cause the package dependencies to be built regardless of whether they were selected using the menu.

10

Example

This example is taken from chan_pika.mk. It shows that chan_pika depends on both the asterisk and hmp packages. It cannot co-exist with the grandprix package so the second TARGETS line removes the grandprix package from the list of targets to build.

Note that if another package with no relationship to chan_pika includes grandprix in its list of targets, it

may still be included if the other package's .mk is parsed after chan_pika.mk. To avoid this, ensure that the entries in Config.in for your package specifiy the menu level dependencies correctly such that this package cannot be selected on the menu. Refer to **Adding Your Package to the Menu (pg. 59)** for details.

```
ifeq ($(strip $(PADS_PACKAGE_CHAN_PIKA)),y)
TARGETS+=asterisk hmp chan_pika
TARGETS-=grandprix
endi
```

10.2 Adding Your Package to the Menu

This section describes the basic information required to add a package into the overall PADS menu. The file <Your PADS Path>/config/Kconfig-language.txt describes other useful details.

Two files are used to implement the overall package menu:

- <Your PADS Path>/Config.in
  - Specifies the parameters for adding the package to the menu.
- <Your PADS Path>/package/Config.in
  - Specifies the complete list of packages that are included in the "Package Selection for the Target" menu displayed when "make menuconfig" is run. It also defines the menu subsections.

Creating the Menu Entry

The file Config.in for each package defines the menu entry and menu dependencies. The following should be specified in the file:

- The PADS internal package name.
- The string shown on the menu.
- The "depends" entry which indicates the relationship to other packages. This entry may be omitted if the package has no relationships.
- The "default" entry indicates whether the package is selected in the menu by default.
- The "help" entry indicates the information that will be displayed when <Help> is selected while the cursor selection is on the package menu item.

Dependencies

Dependencies are specified by a "depends" entry. Specifying dependencies in this manner controls whether the given menu item will be displayed, based on whether the required packages are selected. Multiple dependencies can be specified as well as exclusions. It is important to specify dependencies at the menu level to ensure that packages that are mutually exclusive cannot both be selected.

Entry	Result
depends PADS_PACKAGE_X	If package X is not selected, the package will not be displayed in the menu and therefore cannot be selected. Example: asterisk-gui/Config.in • The asterisk-gui package requires the asterisk package and therefore, cannot be selected without the asterisk package.
depends PADS_PACKAGE_X && PADS_PACKAGE_Y	Both package X and Y must be selected for the package to be displayed and available for selection from the menu • Example: chan_pika/Config.in Both Asterisk and Zaptel Clocking are required and therefore, the channel driver will not be displayed and cannot be selected without both of these packages selected.
depends PADS_PACKAGE_X || PADS_PACKAGE_Y	Either package X or Y must be selected for the package to be displayed and available for selection from the menu.
depends !PADS_PACKAGE_X	If package X is selected, the package cannot be selected and will not be shown on the menu or will be removed from the menu when package X is selected. • Example: grandprix/Config.in Selecting the chan_pika package will remove the grandprix package from the menu and therefore the grandprix package cannot be selected

Adding the Package to the Main Menu

The file package/Config.in must contain an entry for your package so that it can be selected form the "Package Selection for Target" menu. The package may be added into the main menu, you may add a new section to the menu or you may create a new submenu using the menu/endmenu keywords. Refer to the examples below.

10

Example

The following is taken from the Config.in for chan_pika. Note that it depends on three different packages and that it is included by default.

```
config PADS_PACKAGE_CHAN_PIKA
```

```
    bool "PIKA Channel Driver for Asterisk"
    depends on PADS_PACKAGE_ASTERISK && PADS_PACKAGE_ZAPTEL &&
PADS_PACKAGE_HMP
    default y
    help
      Enables Asterisk to use the Warp hardware:
      - FXO, FXS and BRI ports
      - audio line in and line out ports
```

The following example is taken from the grandprix Config.in file. The "depends" entries indicate that the HMP package must be present and that it cannot co-exist with the CHAN_PIKA package. The example above shows that chan_pika is included by default, therefore, in the default view of the menu, PIKA GrandPrix will not appear. PIKA Channel Driver for Asterisk must be unselected for PIKA GrandPrix to appear.

```
config PADS_PACKAGE_GRANDPRIX
    bool "PIKA GrandPrix"
    depends on !PADS_PACKAGE_CHAN_PIKA
    depends on PADS_PACKAGE_HMP
    default n
    help
      PIKA GrandPrix (GP) is a software layer on top of the HMP low-level
      API that makes it easier and faster for designers to develop user
      applications based on PIKA hardware and software.
```

The following examples are taken from the menu "Package Selection for the Target" for which the layout is defined in the file package/Config.in.

This section is shown in the main menu, but is delineated in its own section of the menu which is indicated by the "comment" keyword.

```
comment "PIKA Drivers and SDKs"
source "package/hmp/Config.in"
source "package/grandprix/Config.in"
source "package/lcdlib/Config.in"
```

This portion of the file defines a submenu for utilities which is indicated by the use of the "menu" and "submenu" keywords.

```
menu "Utilities"
source "package/warploader/Config.in"
source "package/gdb/Config.in"
source "package/e2fsprogs/Config.in"
endmenu
```

11 Additional PADS Makefile Rules to Build Software

In addition to the makefile rules for displaying the package selection menu and for building the software, a number of other rules are available.

Rule	Description
make menuconfig	Displays the main package selection menu.
make defconfig	Selects the default set of packages without displaying a menu.
make oldconfig	Preserves the existing package selections and displays a text-based menu to select new package options. This may be useful, for example, when upgrading to a new version of PADS and you wish to start with your existing package selections. If make menuconfig or make defconfig have not previously been executed, this will present a text-based version of the menu selections.
make	Builds all the packages selected.
make clean	Executes the make clean rule for all the packages selected. Note that if you have unselected any packages from the menu the make clean rule for those packages will not be executed.
make dirclean	Executes the make dirclean rule for all the packages selected. Additionally it will remove the following directories: • <Your PADS path>/build_warp • <Your PADS path>/images • Any persistent directories This rule allows you to start fresh with a new build using previously downloaded source code for all previously selected packages.
make image	This will build the ramdisk and persistent file system images based on the files in the directory <Your PADS path>/build_warp/root. Note that the kernel image is built when "make" is executed. Refer to section **Creating Software Images (pg. 67)** for a complete description.

make <package rule>	This will execute the main package rule defined in the <package>.mk file. This package specific rule can be used regardless of whether the package has been selected in the menu. This may be useful if you need to recompile your package/application when there have been no other changes to other packages. Note that it will not build any of the package dependencies specified on the TARGETS line. For example, "make chan_pika" will not build the asterisk package.
make <package rule>-clean	This will execute the clean rule only for the specified package. This package specific rule can be used regardless of whether the package has been selected in the menu. This may be useful if you need to perform a fresh compile of only your package/application when there have been no other changes to other packages. For example, "make chan_pika-clean" will execute directly the clean rule defined in chan_pika.mk.
make <package rule>-dirclean	This will execute the dirclean rule only for the specified package. This package specific rule can be used regardless of whether the package has been selected in the menu. This may be useful if you want to start with a fresh copy of your source code. For example, "make chan_pika-dirclean" will execute directly the dirclean rule defined in chan_pika.mk. Typically, this will remove the package directory from the build_warp directory, however, this will not normally delete the downloaded package source in the directory <Your PADS path>/dl.

12 Using Flash Memory to Run Your Application

Software is typically run from flash memory during testing once initial development has been completed and when the software is ready for deployment. This section provides information about how to write software to flash and other important information about using the flash memory.

- **Flash Memory Partition Layout (pg. 65)** - Describes the flash chips and flash memory layout.
- **Creating Software Images (pg. 67)** - Describes how to build images.
- Three mechanisms are available to update the main flash chip (NAND):
  - **Using the Autoflash Feature (pg. 67)**
  - **Writing Software Images to Flash Using the Warploader (pg. 69)**
  - **Writing Software Images to Flash Using U-Boot (pg. 72)**
- **Updating U-Boot and the FPGA (pg. 73)** - Describes how to build and update the software in the special purpose flash chip (NOR).

12.1 Flash Memory Partition Layout

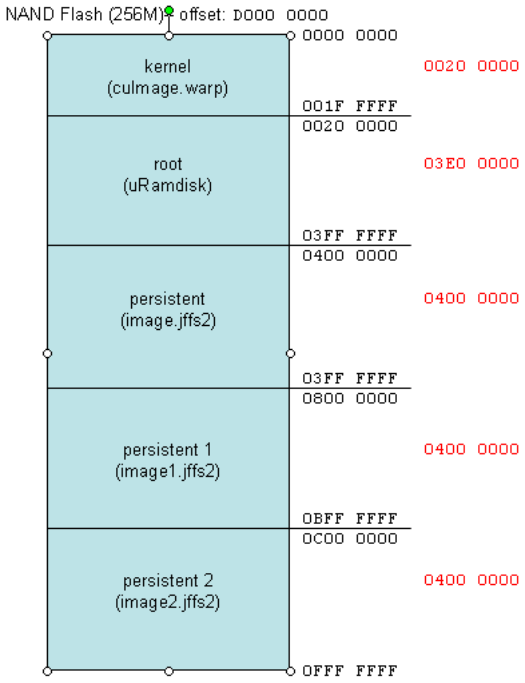
There are two flash memory chips on the appliance. The following sections show the layout of each flash chip and provide information about updating the software in each chip.

12.1.1 NAND Flash

This flash memory is 256 megabyes in size and is used for software (user applications and data) that can be built through PADS.

- kernel (2 Megabytes)
- ramdisk (62 Megabytes)
- persistent file system (64 Megabytes)
- persistent 1 and persistent 2 are intended for user-defined persistent storage (64 Megabytes each)

The kernel and the ramdisk are uncompressed and loaded from flash memory into RAM as part of the boot sequence. The persistent storage sections are not loaded into RAM, files are accessed directly from the flash memory for both reading and writing.



12.1.2 Tracking NAND Writes

Flash memory has a limited number of write-erase cycles and an application that frequently writes to flash may reduce the lifespan of the NAND flash chip. A utility is provided to track the writes to the NAND flash and can be used to monitor excessive or rapidly increasing amounts of data written to flash which may indicate a problem with an application.

To view the amount of data written, enter the following at the Linux prompt on the appliance:

```
cat /proc/driver/ndfc
```

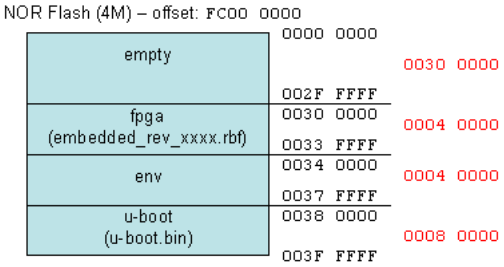
This value is reset to zero between reboots.

12.1.3 NOR Flash

This flash memory is 4 megabytes is size and contains the software required to boot the appliance. It cannot be used by developers for any other purpose.

- U-Boot (512 Kilobytes)
- U-Boot environment variables (256 Kilobytes)

- FPGA (256 Kilobytes)



12.2 Creating Software Images

To create software images for the ramdisk and the persistent file systems, enter the command "make image" on your development computer from the directory <Your PADS path>.

The following images will be created in <Your PADS path>/images:

- cuImage.warp (kernel)
- uRamdisk (ramdisk)
- image.jffs2 (persistent filesystems

If you have used the additional persistent file systems (refer to **Using the Additional Persistent Flash Memory (pg. 44)**), the following images will images will also be created:

- image1.jffs2 (persistent 1)
- image2,jffs2 (persistent 2)

The kernel and ramdisk images are compressed at build time and uncompressed when loaded into RAM during the boot sequence.

12.3 Using the Autoflash Feature

A USB key or SD card can be used to automatically write images to the flash memory when the media is inserted into the system, either before booting the system or while the system is running. To use this feature, it must be enabled in the software running on the appliance (refer to **Advanced Options Menu (pg. 40)** for information about enabling this feature for either USB or SD) These options are enabled by default. The **warploader (pg. 33)** must also be included in the software running on the appliance. The

factory default software on the appliance contains the necessary features to use this functionality.

To use the autoflash feature, create a file called "autorun" in the top level directory on the media that you would like to use (either USB or SD).

A sample autorun file is included with PADS in the directory <Your PADS path>/package/autoflash. The sample copies new kernel, ramdisk and persistent file system images into flash. The sample assumes that the files have the following names or you must edit the script.

File Type	File Name
autorun	autorun (mandatory name)
kernel	cuImage.warp
ramdisk	uRamdisk
persistent	image.jffs2

To use the sample:

1. Copy the images and the autorun script to the appropriate media.
   - Ensure that the media has sufficient space for all the files.
   - Ensure that the autorun script has execute permission enabled.
2. Insert the media into an appliance that is running, or insert it before system startup and then power up the appliance.
   - The media will be auto-mounted in the appliance file system and the autorun script will be executed.
   - Once the script completes, the service that executes the autorun script is disabled and re-inserting the media without rebooting will not replace the images again.
3. Remove the media and reboot the appliance by pressing the reset button.
   - The sample script unmounts the persistent file system and therefore, most functions on the appliance will not work until you reboot the system.

If the autorun script contains instructions to replace the persistent file system, it will fail if an application (such as httpd) has files open on the persistent file system. The script should include instructions to stop such services. The sample script includes the following steps to ensure that there are no files open on the persistent file system:

- stops httpd
- stops tftpd
- logs out any users

WARNING:	The autorun script can contain any set of instructions and therefore, anyone with physical access to the appliance could insert a USB key or SD card with instructions that could cause the appliance to malfunction.

This utility assumes that the USB key is formatted for a FAT file system, which is default when USB keys are purchased.

The file system on the SD card can be either ext2 or FAT. The autoflash utility will mount the SD card with the correct file system, regardless of the entry in /etc/fstab.

12.4 Writing Software Images to Flash Using the Warploader

The warploader is a tool that allows you to write software into flash memory while the appliance is running. The tool provides a single step to replace software, there is no need to return to the U-Boot prompt and no serial connection is required. However, you require network access to the appliance using SSH (server software is included in the factory default software load). The warploader is included in the default software shipped with the appliance.

The kernel, ramdisk, and persistent file system images can all be replaced in flash memory, either with images you have built using PADS or with images downloaded from PIKA's website: **http://www.pikatechnologies.com/appliancedownloads**.

The basic steps are:

1. Create the images by running "make image" from the top level PADS directory, <Your PADS path>.
2. Copy the images to the appliance.
3. Execute the appropriate warploader command for the image type.
4. Reboot the appliance and enter the command "run nand_boot" from the U-Boot prompt.

These step are explained in more detail below.

To write software to flash using the warploader, the software image must be located in the temporary file system (ramdisk) on the appliance. The image can be transferred to the appliance using a file transfer protocol such as SCP or TFTP. Depending on the protocol used, an image can be "pushed" to the appliance from your development machine or the appliance can "pull" it from your development computer.

For example, to retrieve the image file from the /tftpboot directory on your development computer, enter the following at the Linux prompt **on the appliance**:

```
tftp -g -r <filename> <ip address of your development computer>
```

To use SCP to transfer an image from your development computer to the appliance, enter the following command at the Linux prompt **on your development computer**:

```
scp <Your PADS Path>/images/<filename> root@<ip address of the appliance>:/tmp
```

The following commands assume that you have copied the images to /tmp on the appliance.

NOTE:	/tmp is limited to 36 megabytes to avoid using unlimited run-time memory. If the size of the images exceeds the available space, a message will be displayed "No space left on device" when attempting to transfer the image. • If multiple files are using the total available space, delete the image file after it has been written to flash before transferring the next image. • If a single file is greater than 36 megabytes, it may be necessary to increase the size of /tmp. Refer to section **File System Layout (pg. 74)** for more information.

You may access the appliance either from your serial client or SSH. The following warploader commands are executed at the Linux prompt **on the appliance**.

The basic usage is:

```
warploader -p <partition name> file name
```

Image type	Partition name	Warploader command
kernel	kernel	warploader -p kernel /tmp/cuImage.warp
ramdisk	root	warploader -p root /tmp/uRamdisk

persistent file system	persistent	**Unless your system is running with the persistent file system on NFS, you must unmount the persistent file system before replacing it. The warploader will return an error if you have not unmounted the filesystem. If there are services running that have files open on the persistent file system, you will need to stop those services before unmounting.** For example, if httpd is running on the system, it will have files open on the system. The process tftpd will also have files open. To kill the processes: • killall httpd • killall tftpd Ensure that any users logged in do not have files open in persistent and that the current directory for any user is not in persistent. For example, the root user's home directory is /persistent/root. The root user should change to another directory before upgrading the persistent file system. Then continue with updating the persistent file system. umount /persistent warploader -p persistent /tmp/image.jffs2
persistent 1	persistent1	umount /persistent1 warploader -p persistent1 /tmp/image1.jffs2
persistent 2	persistent2	umount /persistent2 warploader -p persistent2 /tmp/image2.jffs2

The size of the image is verified to ensure that it will fit into the space allocated in flash for the image type.

Ensure that the U-Boot environment variable ***bootcmd*** is set to **"run nand_boot"** (the factory default setting) and enter the command 'reboot' at the Linux prompt to boot using the new images.

If you have previously modified the value to run from NFS, change the value of ***bootcmd*** by entering the following commands at the U-Boot prompt.

```
=>setenv bootcmd run nand_boot
=>saveenv
```

Note that if any of the images are corrupt, are the wrong image for the specified partition (e.g. writing the ramdisk image to the kernel partition) or if the system is interrupted in any way during the update (pressing the reset button, disconnecting the power), your system will not boot and you must return to U-Boot to repair your system by updating the flash (refer to section

Writing Images to Flash Using U-Boot (pg. 72)) or boot instead from NFS.

12.5 Writing Software Images to Flash Using U-Boot

To use U-Boot to write software images into flash memory, you must connect the appliance to your development computer using the serial cable. Return to the U-Boot prompt by rebooting the appliance and pressing any key during the boot sequence as described in **Installing and Running the Software (pg. 13)**. Ensure that the U-Boot environment variables are set as described.

The "update" command is provided in U-Boot to burn images to flash. Enter "help update" at the U-Boot prompt for information.

You must copy the files to your /tftpboot directory on your development machine before using the "update" command. The following instructions assume that you have done so.

Image type	Command
kernel	update kernel cuImage.warp
ramdisk	update ramdisk uRamdisk
persistent file system	update persistent image.jffs2
persistent 1	update user1 image1.jffs2
persistent 2	update user2 image2.jffs2

Note that if any of the images are corrupt, are the wrong image for the specified partition (e.g. writing the ramdisk image to the kernel partition) or if the system is interrupted in any way during the update (pressing the reset button, disconnecting the power), your system will not boot and you must return to U-Boot to repair your system by updating the flash or boot instead from NFS.

Ensure that the ***bootcmd*** environment variable is set to **"run nand_boot"** and enter the command 'reset' at the U-Boot prompt to boot the appliance using the new images.

12.6 Updating U-Boot and the FPGA

The FPGA and U-Boot images in the NOR should not be replaced except under the direction of PIKA Technologies, either through **Customer Care (pg. 2)** or by written instructions.

PADS cannot build the FPGA, PIKA Technologies supplies the image file. To build the U-Boot software using PADS, from <Your PADS path>, enter the following command:

```
make uboot
```

The file u-boot.bin will be created in <Your PADS path>/images.

To replace the FPGA and U-Boot using the warploader, you must transfer the images to the appliance using a file transfer protocol such as SCP or TFTP. The following instructions, which are executed on the appliance, assume that you have copied your images into /root on the appliance:

Image type	Warploader Command
U-Boot	warploader -p u-boot /root/u-boot.bin
FPGA	warploader -p fpga /root/embedded_rev_<fpga version>.rbf

The warploader validates the U-Boot image file to ensure that it has a valid file format.

To replace the FPGA and U-Boot from U-Boot, copy the images to /tftpboot on your development computer and enter the following commands at the U-Boot prompt:

Image type	U-Boot Command
U-Boot	update uboot u-boot.bin
FPGA	update fpga embedded_rev_<fpga version>.rbf

13 Advanced Topics

The following sections contain important information for certain types of applications you may be developing and for more advanced system setup of the appliance.

13.1 File System Layout

Software ramdisks use the normal RAM in main memory as if it were a partition on a hard drive. The appliance ramdisk contains the root file system which is loaded into RAM at boot time and therefore, changes are lost upon reboot.

Three temporary file systems are created at the following mount points in the ramdisk to limit the amount of RAM that can be used by these commonly accessed directories. Space in RAM is only used when files are written to these directories (space is not set aside up front). Refer to the file /etc/fstab for specific settings.

- /var/run (limited to 1 M)
- /var/log (limited to 1 M)
- /tmp (limited to 36 M)

To create a new temporary file system, add an entry to /etc/fstab. For example, to make /var/tmp a temporary file system, add the following entry:

```
tmpfs        /var/tmp      tmpfs   size=1m        0 0
```

To change the size of an existing file system, the size parameter may be adjusted accordingly. Size can be specified in either megabytes (m) or kilobytes (k).

Entries in /etc/fstab also specify the file system type and whether the file system will be automatically mounted at system startup. File systems indicated as "noauto" will not be mounted automatically. Note that the device must be present at system startup in order to mount automatically. If the device is not present, there will be an error shown on the console. If the device is subsequently inserted after system startup, it will still need to be mounted manually. Note that some devices may take too long to mount and automounting at system startup may still fail.

The following shows the default entries in /etc/fstab. Four different types of file systems are shown:

- tmpfs (described above)
- ext2 (refer to section **ext2 File System Utilities for USB and SD Media (pg. 32)**)
- vfat (commonly used for Windows file systems)
- jffs2 (Journaling Flash File System, version 2, used for all persistent flash partitions)

```
tmpfs                /var/run      tmpfs   size=1m       0 0
tmpfs                /var/log      tmpfs   size=1m       0 0
tmpfs                /tmp          tmpfs   size=36m      0 0
/dev/mmcblk0p1 /mnt/sd       ext2     noauto          0 0
/dev/sda1           /mnt/usb     vfat     noauto          0 0
/dev/mtdblock7   /persistent1   jffs2    auto            0 0
/dev/mtdblock8   /persistent2   jffs2    auto            0 0
```

Changes to /etc/fstab do not take effect until the next reboot.

NOTE:	If there is an SD card inserted at system startup, there is an entry in /etc/rc.S that mounts it, even though the default entry in /etc/fstab is "noauto". The custom appliance SD driver handles the case where the SD is not present at system startup and prevents errors from showing on the console. If the autoflash feature is present on the system (refer to **Using the Autoflash Feature (pg. 67)**), the SD will be mounted automatically if it is inserted after system startup and will remain mounted even after the autoflash script has executed. Adding an entry to /etc/rc.S will not work for the USB as it does for the SD; if it is not present at system startup, an error will occur. It has the same effect as changing the entry in /etc/fstab to "auto". Additionally, the autoflash feature does not leave the USB mounted when it is finished.

When running from flash, persistent flash partitions are mounted into the file system using the mtdblock devices. These partitions are not loaded into memory and all reads and writes access the flash directly. The following mtd blocks are used for the persistent flash partitions:

- persistent is mapped to mtdblock6
- persistent1 is mapped to mtdblock7
- persistent2 is mapped to mtdblock8

To view the current mtd block table, enter the command "cat /proc/mtd" at the Linux prompt on the appliance.

13.2 Logging

On a regular Linux operating system, run-time logs are sent to the /var/log directory. On the appliance, this directory is part of the ramdisk and therefore, the contents are not preserved across system reboots. The /var/log portion of the file system is configured as a temporary file system and its size is limited to 1 Kb to avoid consuming excess RAM on the appliance. Note that enabling debug logs may quickly consume the available space, limiting the amount of information captured.

By default, all Asterisk logs are written to /var/log/asterisk. Logs for PIKA software are directed to syslog which, by default, is /var/log/messages on the appliance.

The following sections describe some options for directing logs to alternate locations so that they are available should the system reboot unexpectedly or if the logs will consume more than the allocated space in /var/log. Log setup as described below is done at initialization time.

Using Syslog to Send Logs to a Remote System

Syslog is a standard for forwarding log messages in an IP network. The term "syslog" is often used for both the actual syslog protocol, as well as the application or library sending syslog messages. The appliance can use syslog for packages that support its use. Using syslog for capturing log information is recommended to avoid consuming memory on the appliance. If syslog is configured to send logs to a remote system, logs are sent to /var/log/messages on the remote host.

By default, Asterisk logs are sent to /var/log/asterisk. The contents of this directory will not be preserved across reboots. To direct Asterisk logs to syslog, change the appropriate setting in /etc/asterisk/logger.conf.

By default, PIKA HMP and PIKA GrandPrix (either stand-alone or from within the PIKA Asterisk channel driver) logs are directed to syslog.

The following environment variables are set during the initialization sequence in the file /persistent/etc/localenv:

Component	Value
HMP	export PKH_LOGS_DIR=SYSLOG
GP	export PKX_LOGS_DIR=SYSLOG

To enable syslog:

On the remote host where the logs will be stored, execute the following:

```
syslogd -m 0 -r
```

On the Appliance, execute the following:

```
syslogd -R <remote host IP>
```

If logs will be permanently redirected to a remote host, ensure that syslog is started at boot time on both the remote host and the appliance.

The following syslog levels can be set for PIKA HMP and PIKA GP when exporting the log environment variable. For example:

```
export PKX_LOGS_DIR=SYSLOG LOG_INFO
```

Log Level	Meaning
LOG_EMERG	system is unusable
LOG_ALERT	action must be taken immediately
LOG_CRIT	critical conditions
LOG_ERR	error conditions
LOG_WARNING	warning conditions
LOG_NOTICE	normal, but significant, condition
LOG_INFO	informational message
LOG_DEBUG	debug-level message The use of debug level logging will significantly degrade the performance of the appliance.

Using the SD Card for Logging

To direct logs to the SD card for permanent storage, the following changes are required. The SD card **must** be inserted at system startup.

PIKA HMP:

In the file /persistent/etc/localenv, change the following line:

```
export PKH_LOGS_DIR=SYSLOG
```

to

```
export PKH_LOGS_DIR=/mnt/sd/<your log directory name>
```

PIKA GrandPrix:

In the file /persistent/etc/localenv, change the following line:

```
export PKX_LOGS_DIR=SYSLOG
```

to

```
export PKX_LOGS_DIR=/mnt/sd/<your log directory name>
```

You must reboot for these changes to take effect.

Asterisk:

In /etc/asterisk/asterisk.conf, change the following variables as shown below:

```
astspooldir =>/mnt/sd/<your log directory name>
astlogdir => /mnt/sd/<your log directory name>
```

If logs are sent to a subdirectory of /mnt/sd, the permissions for that directory must be set to so that the "asterisk" user can write to them. There are two options:

1. Change the permissions so that all users have read, write and execute permissions by executing the following commands at the Linux prompt on the appliance after you have created the subdirectory:

```
chmod 777 /mnt/sd/<your log directory name>
```

2. Change the directory ownership so that the "asterisk" user owns the directories by executing the following commands at the Linux prompt on the appliance after you have created the subdirectory:

```
chown -R asterisk:asterisk /mnt/sd/<your log directory name>
```

Note that any files, including voice mail files, that are normally sent to any subdirectory specified by the *astspooldir* variable will now be sent to the SD. Restart Asterisk to activate the changes.

13.3 Network Settings

Network information for the appliance can be set either in U-Boot or in the file /etc/networking.conf. This section explains the relationship and interactions between them and when to use each of them.

When the U-Boot environment variable ***ipaddr*** is set to '**dflt**' (the factory preset value), network settings are based on the information in /etc/networking.conf. Changes to the network settings should be made by editing the file /etc/networking.conf.

During development, the network information in U-Boot must be set in order to boot from NFS and to replace software images in the flash memory using U-Boot. If the U-Boot environment variable ***ipaddr*** does not have a valid IP address, NFS will not work; the system will fail to boot and will return to the U-Boot prompt. The U-Boot settings can only be changed by returning to the U-Boot prompt (when connected via the serial port). No software running on the appliance will change U-Boot settings.

Network settings are always used entirely from either /etc/networking.conf or from U-Boot; a combination of the information is never used.

Network configuration information should be set in /etc/networking.conf for production systems and the U-Boot environment variable ***ipaddr*** should be set to '**dflt**'. The following table shows the settings in /etc/networking.conf and their default values.

Setting	Usage	Factory Preset Value
IP_LAN	IP address to configure on the network interface	192.168.1.80
NETMASK_LAN	Netmask to configure on the network interface (relative to IP_LAN)	255.255.255.0
GATEWAY_LAN	IP Address of the default route	192.168.1.1

DHCP_LAN	Use DHCP to configure networking (yes|no). IP_LAN must not be set for DHCP_LAN=yes to be honored	no
DNS_LAN	IP address of the DNS server to use	none
HOSTNAME	Name by which this host should be known	warp
DOMAIN_LAN	Domain name of the local network	none, entry not present in the file by default

The following U-Boot environment variables for network configuration must be set for development.

Environment Variable	Usage	Factory Preset Value
gatewayip	IP Address of the default route	0.0.0.0
netmask	Netmask to configure on the network interface (relative to ipaddr)	255.255.255.0
ipaddr	IP address to configure on the network interface	deflt
serverip	IP address of your development computer to use for NFS and TFTP	0.0.0.0

While there is a U-Boot environment variable called hostname, it is not used once the system has finished booting.

NOTE:	Ensure that the U-Boot environment variable ***ipaddr*** is set to '**deflt**' for production systems.

The file /etc/HOSTNAME will be used for the hostname in the following situations:

- the U-Boot environment variable ***ipaddr*** is set to 'deflt' and there is no value for HOSTNAME in /etc/networking.conf
  - to use a different hostname, change the value in /etc/networking.conf
- the U-Boot environment variable ***ipaddr*** is a valid IP address
  - to use a different hostname, change the value in /etc/HOSTNAME

13

13.4 Displaying Information on the LCD

The package "PIKA LCD Library API" provides an interface for applications to display information on the appliance LCD. At this time, the API only supports a single application using the LCD display. If multiple

applications attempt to write to the display, each application will overwrite other applications' information.

By default, the astmanproxy application uses the LCD to display Asterisk call status information. If another application wishes to use the LCD, Asterisk must not be included in the software running on the appliance.

The "PIKA HMP IVR Sample Application" (refer to **Samples (pg. 34)** for information) is a non-Asterisk sample application that provides an example of updating the LCD with call status information.

The PIKA HMP package must be included to use the LCD library.

To prepare the LCD for use, the user application uses the function **PK_LCD_Open (pg. 109)**. The function returns a handle to use for all subsequent function calls relating to the LCD. Configuration information can be set using the function **PK_LCD_SetConfig (pg. 111)**. The function **PK_LCD_GetConfig (pg. 107)** should always be called prior to calling this function. This guarantees that all fields (including fields added in later releases of this software) are set to proper default values.

Options that can be configured:

orientation

- normal or reversed

mode

- text or bitmap

shift time

- interval in milliseconds to use when scrolling text across the LCD, used only in text mode

blink time

- interval in milliseconds between blinks, used only in bitmap mode

brightness

- brightness of the LCD background

The default display mode is text. Bitmap mode is typically used to display images, while text mode is used to display simple text strings.

To display a text string on the LCD, the user application uses the function **PK_LCD_DisplayString (pg. 104)**, specifying the handle returned from the function **PK_LCD_Open (pg. 109)** and the following information about the string:

string - pointer to the character buffer containing the string to be displayed

length - string length
line number - line on the LCD on which the string should be displayed

If the length of the string is greater than the width of the LCD, the string will scroll across the screen. In text mode, the previous contents of the display are cleared before displaying the new string.

To display a bitmap on the LCD, the user application uses the function **PK_LCD_DisplayBitmap (pg. 103)**, specifying the handle returned from the function **PK_LCD_Open (pg. 109)** and the following information about the bitmap:
region
- starting horizontal and vertical pixels
- horizontal and vertical sizes

bitmap
- the pattern to display

If there is already content in the section of the LCD display specified in the region parameter, calling the display function will overwrite the contents. To start with a blank display, the user application can use the function **PK_LCD_Clear (pg. 92)** before calling the display function.

The functions **PK_LCD_SetBlink (pg. 110)** and **PK_LCD_UnsetBlink (pg. 113)** can be used in bitmap mode to cause a region of the LCD to blink and stop blinking.

Once the user application has finished using the LCD, the function **PK_LCD_Close (pg. 95)** should be used to free the associated resources. It is a good practice to use the function **PK_LCD_Clear (pg. 92)** to clear the display before closing.

To receive notifications when the LCD button is pressed, the user application must attach an event handler to the LCD handle using the **PK_LCD_SetEventHandler (pg. 112)** and specify the following information:
handle - LCD handle returned from **PK_LCD_Open (pg. 109)**
callback - the function that will be called when the event is raised
user data - information specific to the user application

When the event **PK_LCD_EVENT_LCD_BUTTON_PRESSED (pg. 104)** is raised, paramter p0 indicates the number of times the button was pressed.

When the application no longer requires information about button presses, it should call **PK_LCD_ResetEventHandler (pg. 109)** to deregister the callback function.

If the user application does not need information about button presses, a callback function does not need to be registered.

For detailed information about the LCD API, refer to **Appendix A - LCD API Reference (pg. 91)**.

13.5 Using the Persistent File Systems from Flash

The persistent file system resides on your development computer by default because it is expected that developers will use NFS for primary development. In this case, the following directories are used for the persistent data:

- <Your PADS path>/build_warp/root/persistent
- <Your PADS path>/build_warp/root/persistent1
- <Your PADS path>/build_warp/root/persistent2

When running from NFS, there may be a need to use the persistent file systems in flash, for example, when debugging a system from the field with its current configuration settings. In this case, the option "Persistent File System in Flash" can be selected using the **Advanced Options Menu (pg. 40)** when building the ramdisk. All persistent file systems, including the additional persistent partitions, will be accessed from flash instead of NFS.

Note that this option has no effect when the ramdisk is run from flash; the ramdisk in flash will always access the persistent file systems in flash.

13.6 U-Boot Environment Variables

The following table summarizes the U-Boot environment variables and their use. If it becomes necessary to revert the U-Boot environment to the factory preset values, the following command can be executed at the U-Boot prompt:

```
=>defenvs
```

Variable	Value	Default Value
def_env	PIKA internal use, do not change	1
postdelay	Number of seconds to wait for "p" to be pressed to enter POST (Power On Self Tests). If "p" is not pressed before the specified number of seconds expires, booting will continue.	1
bootcmd	Indicates whether the appliance will boot from flash or NFS. Valid values: • run nand_boot • run net_nfs	run nand_boot
bootdelay	Number of seconds to wait before continuing to boot after the FPGA is finished loading.	3
baudrate	Serial port baudrate	115200
loads_echo	If set to 1, all characters received during a serial download are echoed back.	1
preboot	Message to display before continuing to boot after the FPGA is finished loading.	Empty line
netdev	Name of the network device	eth0
hostname	Hostname used for the appliance. Not used after the system has finished booting.	warp
post_dma_lb_loops	Number of times the DMA POST loopback test will run. Do not change this value.	10
nfsargs	Arguments to pass to the kernel when booting from NFS. Do not change this value.	setenv bootargs root=/dev/nfs rw nfsroot=${serverip}:${rootpath}
ramargs	Arguments to pass to the kernel when booting from flash memory. You may need to change the value of ramdisk_size if you have increased ROOTFS_SIZE in the main Makefile.	setenv bootargs root=/dev/ram rw **ramdisk_size=130000**

addip	Network settings to pass to the kernel at boot time. This information is appended to the existing bootargs value. Do not change this value	setenv bootargs ${bootargs} ip= ${ipaddr}:${serverip}:${gatewayip}: ${netmask}:${hostname}:${netdev}:off panic=1
addtty	Serial port communication settings to pass to the kernel at boot time. This information is appended to the existing bootargs value. Do not change this value	setenv bootargs ${bootargs} console=ttyS0,${baudrate}
bootfile	Default file name of the kernel image to load from TFTP, which is used when booting from NFS.	cuImage.warp
net_nfs	Command to boot from NFS, do not change this value.	tftp 200000 ${bootfile};run nfsargs addip addtty;bootm
load_nand_kernel	Instructions used when booting the kernel from flash memory. Do not change this value.	nand read.jffs2 2000000 0 180000
load_nand_ramdisk	Instructions used when reading the ramdisk from flash memory into RAM. The parameters for nand read.jffs2 are the following: • address in RAM to start writing • address in NAND to start reading • number of bytes (hex) to read from NAND The value for the last parameter may need to be changed if compressed ramdisk image size is larger than 48M.	load_nand_ramdisk nand read.jffs2 2200000 200000 **3000000**
nand_boot	Command to boot from flash memory, do not change this value.	run ramargs addip addtty load_nand_kernel load_nand_ramdisk;bootm 2000000 2200000
serverip	IP address of your development computer	Not set
ipaddr	IP address of the appliance	deflt
gatewayip	gateway IP for your local network	0.0.0.0
netmask	network mask for your local network	255.255.255.0
rootpath	path to the NFS mount point on your development machine	/opt/eldk/ppc_4xx

ethact	Active Ethernet port. The appliance has only one Ethernet port, do not change this value.	ppc_4xx_eth0

13.7 Retrieving System Identifcation Information

When writing applications, information about the system hardware, including the serial number, can be retrieved using the PIKA HMP API function PKH_BOARD_GetInfo. The PIKA HMP package must be included to use the API.

For more information about using the PIKA HMP SDK, refer to the documentation available on the PIKA website at **www.pikatechnologies.com**. Under the "Support and Downloads" tab, "Documentation" menu, select "Current SW Releases".

14 Frequently Asked Questions

This section answers some typical user questions.

14.1 How do I run software from NFS?

1. Download PADS from **http://www.pikatechnologies.com/appliancedownloads** to your Linux development computer.
2. Build PADS:
   1. Enter the command "make menuconfig" from the root PADS directory, <Your PADS path>.
   2. Enter "make" from the root PADS directory, <Your PADS path>.
      - The folder <Your PADS path>/build_warp/root will contain the mount point to boot the software.
3. Ensure a tftp server is installed on your development computer (see **Setting up TFTP and NFS (pg. 9)** for information).
4. Copy the file <Your PADS path>/images/cuImage.warp to the /tftpboot folder on your development computer.
5. Ensure that NFS is installed and running on your development computer (refer to **Setting up TFTP and NFS (pg. 9)** for information).
6. On your development machine, modify the file /etc/exports file with the appropriate information.
7. Use the serial cable to connect your development computer to the appliance.
8. Start minicom on your development computer to access the serial console on the appliance. Re-boot the appliance and press a key to enter U-Boot.
9. Once in U-Boot, set the following variables using the 'setenv' and 'saveenv' commands:
   - ipaddr
   - gatewayip
   - netmask
   - serverip (the IP of your Linux development computer)
   - rootpath (<Your PADS path>/build_warp/root on your Linux development computer) (refer to **Running the Software from NFS (pg. 13)** for more information).
10. From the U-Boot prompt, type "run net_nfs" to boot the appliance.

15 Troubleshooting

The following section lists some common problems and solutions. If you have a problem not described in this section, please check the FAQ and Troubleshooting sections on the PIKA Technologies web site **http://www.pikatechnologies.com/appliancedownloads**.

NFS

The following are common NFS errors:

- NFS error -5 indicates that the nfsd service is not started on your development computer. Check that the service is running and configured to start when your development computer is booted.
- NFS error -13 indicates that your NFS path cannot be found. Check that the path in /etc/exports matches the ***rootpath*** environment variable in U-Boot and that the path specified contains a valid mount point. If you changed any information in /etc/exports, run "exportfs -a" on your development computer.

When attempting to start the NFS service, the following error appears:

```
Starting NFS quotas: Cannot register service: RPC: Unable to receive; errno = Connection refused
```

Check if the portmap service is running:

```
#ps ax | grep portmap
```

If not, start the portmap service using the appropriate command for your Linux distribution (e.g. "service portmap start"). Ensure that it is configured to start when the system is booted.

NFS and TFTP

The message "Remote system error - No route to host" may indicate that a firewall is interfering with access to your development computer. To resolve this issue, either disable the firewall or configure the firewall to allow these services. Firewall configuration is beyond the scope of this document.

Note that SELinux must be disabled to use NFS and TFTP.

SSH

The message "dbclient: exited: string too long" when passing an RSA key file to the SSH client (dbclient,

included with the dropbear package) indicates that the authentication keys used are not compatible with dropbear.

Dropbear is a subset of the full SSH functionality. When the SSH client on the appliance is used to access another system running a fully featured SSH server, the private authentication keys supplied by the server require a conversion step to be compatible with dropbear. Execute the following command on the appliance:

```
#dropbearconvert openssh dropbear /path/to/keyFile ~/.ssh/id_rsa.db
```

'make image' Fails

In addition to exceeding the image size set in the top level Makefile of <Your PADS Path>, 'make image' can also fail with the following error:

```
genext2fs -U \
-d build_warp/root \
-b 130000 \
-i 2048 \
images/rootfs.img
genext2fs: invalid option -- U
make: *** [image] Error 1
```

This may occur if you have more than one version of genext2fs on your development computer. The version in <Your PADS Path>/bin should be used. The following entries in the top level Makefile may need to be adjusted if any of the directories listed in PATH contain the executable genext2fs.

```
PATH:= /usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
PATH:= $(PATH):$(TOOLS_BIN)
```

Error Attempting to Unmount Persistent Filesystems

umount: cannot umount /persistent: Invalid argument

This error will appear if you attempt to unmount any of the persistent files systems (persistent, persistent1 persistent2) when you are running with the persistent file systems on NFS. Unmounting is only necessary (for example, to write the a new image to flash using the warploader) if the persistent file systems are in flash.

Asterisk GUI Checkout Fails

When PADS attempts to build the Asterisk-GUI on CentOS4/RedHat ES4 systems, this message may occur:

```
svn: URL 'http://svn.digium.com/svn/asterisk-gui/branches/1.0' doesn't exist
```

This indicates that the version of SVN on your development computer is too old. Version 1.4 or later is required. Use "yum update subversion" to get the latest version.

16 Appendix A - LCD API Reference

The following sections provide reference information for the LCD library API.

Functions

PK_LCD_Clear (pg. 92)	The PK_LCD_Clear function is a utility function that sets the LCD to the blank state both in text mode and bitmap mode. In bitmap mode, it will clear the content of the LCD and cancel all the blink operations set before. In text mode, the content of all line buffers will be cleared.
PK_LCD_Close (pg. 95)	The PK_LCD_Close function closes down the LCD and invalidates its handle.
PK_LCD_DisableLogs (pg. 100)	The PK_LCD_DisableLogs function disables debug logging in PK_LCD.
PK_LCD_DisplayBitmap (pg. 103)	The PK_LCD_DisplayBitmap function displays a bitmap on the LCD in the specified region.
PK_LCD_DisplayString (pg. 104)	The PK_LCD_DisplayString displays a string to the specified line on the LCD.
PK_LCD_EnableLogs (pg. 105)	The PK_LCD_EnableLogs function enables debug logging in PK_LCD.
PK_LCD_ERROR_GetText (pg. 106)	The PK_LCD_ERROR_GetText function returns the name of the status code in a user-provided buffer.
PK_LCD_EVENT_GetText (pg. 107)	The PK_LCD_EVENT_GetText function returns the name of the event in a user -provided buffer.
PK_LCD_GetConfig (pg. 107)	The PK_LCD_GetConfig function retrieves the current configuration settings of the specified LCD.
PK_LCD_GetInfo (pg. 108)	The PK_LCD_GetInfo function retrieves the capability information of the specified LCD.
PK_LCD_Open (pg. 109)	The PK_LCD_Open function allocates a LCD object and returns its handle to the calling routine.
PK_LCD_ResetEventHandler (pg. 109)	The PK_LCD_ResetEventHandler function reset the callback function used to notify events generated by LCD.
PK_LCD_SetBlink (pg. 110)	The PK_LCD_SetBlink function will set a region in the LCD to blink with the speed specified by the displayBlinkTime parameter in the configure struct.
PK_LCD_SetConfig (pg. 111)	The PK_LCD_SetConfig function sets the configuration settings of the specified LCD.
PK_LCD_SetEventHandler (pg. 112)	The PK_LCD_SetEventHandler function sets the callback function for notifying events generated by LCD.
PK_LCD_SetFontLib (pg. 113)	The PK_LCD_SetFontLib function sets the bitmap library of the ASCII characters to replace to the default one.

PK_LCD_UnsetBlink (pg. 113)	The PK_LCD_UnsetBlink function will cancel the blink operation previously set by the **PK_LCD_SetBlink (pg. 110)** function call.

16.1 PK_LCD_Clear

The PK_LCD_Clear function is a utility function that sets the LCD to the blank state both in text mode and bitmap mode. In bitmap mode, it will clear the content of the LCD and cancel all the blink operations set before. In text mode, the content of all line buffers will be cleared.

```
PK_STATUS PK_API PK_LCD_Clear(
    IN TPikaHandle lcdHandle
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_SUCCESS	The function succeeded.

Remarks

None.

16.2 LCD Structures, Unions and Enumerations

Structures

PK_LCD_TLCDConfig (pg. 93)	The PK_LCD_TLCDConfig structure is used by **PK_LCD_GetConfig (pg. 107)** and **PK_LCD_SetConfig (pg. 111)** function to retrieve and set the configuration of the LCD display.
PK_LCD_TLCDInfo (pg. 93)	The PK_LCD_TLCDInfo structure is used by PK_LCD_GetLCDInformation function to retrieve read-only information describing the dimensions of the LCD display in text mode and bitmap mode.
PK_LCD_TLCDRegion (pg. 94)	There are no defaults for this since it is application specific

PK_LCD_TPikaEvent (pg. 94)	The PK_LCD_TPikaEvent structure is the basic vehicle for passing asynchronous data to the user application.

16.2.1 PK_LCD_TLCDConfig

The PK_LCD_TLCDConfig structure is used by **PK_LCD_GetConfig (pg. 107)** and **PK_LCD_SetConfig (pg. 111)** function to retrieve and set the configuration of the LCD display.

```
typedef struct {
  PK_UINT orientation;
  PK_UINT mode;
  PK_UINT shiftTime;
  PK_UINT blinkTime;
  PK_UINT brightness;
} PK_LCD_TLCDConfig;
```

Members

Members	Description
orientation	Determines the shift direction and the orientation of the characters, rotated 180 degrees or normal. Default is **PK_LCD_ORIENTATION_NORMAL (pg. 98)**
mode	Display mode of the LCD, text mode or bitmap mode. Default is **PK_LCD_DISPLAY_MODE_TEXT (pg. 98)**
shiftTime	The interval between shifts in ms, only used in text mode. Default is **PK_LCD_SHIFT_INTERVAL_DEFAULT (pg. 99)** (500 ms)
blinkTime	The interval between blinks in ms, only used in bitmap mode. Default is **PK_LCD_BLINK_INTERVAL_DEFAULT (pg. 97)** (500 ms)
brightness	The brightness of the LCD background lighting. Default is **PK_LCD_BRIGHTNESS_100 (pg. 97)**

16.2.2 PK_LCD_TLCDInfo

The PK_LCD_TLCDInfo structure is used by PK_LCD_GetLCDInformation function to retrieve read-only information describing the dimensions of the LCD display in text mode and bitmap mode.

```
typedef struct {
  struct {
    PK_UINT characters;
    PK_UINT lines;
  } textMode;
  struct {
    PK_UINT width;
    PK_UINT height;
  } bitmapMode;
} PK_LCD_TLCDInfo;
```

Members

Members	Description
textMode	Dimensions related to text mode
characters	The maximum number of characters that can be displayed on one line of the LCD. This number can be larger than the physical width of the LCD. In that case, the string will be scrolled automatically using the shiftTime parameter in the config structure. Default is **PK_LCD_TEXT_LINE_BUFFER_LENGTH (pg. 99)** (40)
lines	The maximum number of lines that can be displayed on the LCD. Default is **PK_LCD_TEXT_LINES (pg. 99)** (2)
bitmapMode	Dimensions related to bitmap mode
width	The maximum number of horizontal pixels. Default is **PK_LCD_BITMAP_WIDTH (pg. 96)** (160)
height	The maximum number of vertical pixels. Default is **PK_LCD_BITMAP_HEIGHT (pg. 96)** (32)

16.2.3 PK_LCD_TLCDRegion

There are no defaults for this since it is application specific

```
typedef struct {
  PK_UINT x;
  PK_UINT y;
  PK_UINT width;
  PK_UINT height;
} PK_LCD_TLCDRegion;
```

Members

Members	Description
x	The start of horizontal position of the region.
y	The start of vertical position of the region.
width	The horizontal width of the region.
height	The vertical height of the region.

16.2.4 PK_LCD_TPikaEvent

The PK_LCD_TPikaEvent structure is the basic vehicle for passing asynchronous data to the user application.

```
typedef struct {
  PK_UINT id;
  TPikaHandle handle;
  PK_TIMESTAMP_MS timestamp;
  PK_VOID * userData;
  PK_UINTPTR p0;
```

```
  PK_UINTPTR p1;
  PK_UINTPTR p2;
} PK_LCD_TPikaEvent;
```

Members

Members	Description
id	The event id
handle	The handle of the object raising the event.
timestamp	The time the event was raised (in milliseconds since the computer was started).
userData	The user data associated with the object (see **PK_LCD_SetEventHandler (pg. 112)**).
p0	The first parameter of the event.
p1	The second parameter of the event.
p2	The third parameter of the event.

16.3 PK_LCD_Close

The PK_LCD_Close function closes down the LCD and invalidates its handle.

```
PK_STATUS PK_API PK_LCD_Close(
    IN TPikaHandle lcdHandle
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_SUCCESS	The function succeeded.

Remarks

This function decreases the use count of the LCD service. If the use count is equal to zero, it frees all resources allocated by the LCD.

16.4 LCD Constants

Macros

PK_LCD_BITMAP_HEIGHT (pg. 96)	Maximum number of vertical pixels in bitmap mode.
PK_LCD_BITMAP_WIDTH (pg. 96)	Maximum number of horizontal pixels in bitmap mode.
PK_LCD_BLINK_INTERVAL_DEFAULT (pg. 97)	Default LCD blink interval in ms.
PK_LCD_BLINK_INTERVAL_MAX (pg. 97)	Maximum LCD blink interval in ms.
PK_LCD_BLINK_INTERVAL_MIN (pg. 97)	Minimum LCD blink interval in ms.
PK_LCD_BRIGHTNESS_0 (pg. 97)	The brightness of the LCD back lighting is 0 percent.
PK_LCD_BRIGHTNESS_100 (pg. 97)	The brightness of the LCD back lighting is 100 percent.
PK_LCD_BRIGHTNESS_25 (pg. 97)	The brightness of the LCD back lighting is 25 percent.
PK_LCD_BRIGHTNESS_50 (pg. 97)	The brightness of the LCD back lighting is 50 percent.
PK_LCD_BRIGHTNESS_75 (pg. 98)	The brightness of the LCD back lighting is 75 percent.
PK_LCD_DISPLAY_MODE_BITMAP (pg. 98)	The display mode of the LCD is bitmap mode.
PK_LCD_DISPLAY_MODE_TEXT (pg. 98)	The display mode of the LCD is text mode.
PK_LCD_EVENT_MAX_NAME_LENGTH (pg. 98)	Maximum length of an event name.
PK_LCD_ERROR_MAX_NAME_LENGTH (pg. 98)	Maximum length of a status code name.
PK_LCD_ORIENTATION_NORMAL (pg. 98)	The orientation of the LCD is normal.
PK_LCD_ORIENTATION_REVERSED (pg. 98)	The orientation of the LCD is reversed.
PK_LCD_REGION_FULL_SCREEN (pg. 99)	Convenience definition for the full screen region
PK_LCD_SHIFT_INTERVAL_DEFAULT (pg. 99)	Default LCD shift interval in ms.
PK_LCD_SHIFT_INTERVAL_MAX (pg. 99)	Maximum LCD shift interval in ms.
PK_LCD_SHIFT_INTERVAL_MIN (pg. 99)	Minimum LCD shift interval in ms.
PK_LCD_TEXT_LINE_BUFFER_LENGTH (pg. 99)	Number of characters in the LCD display line buffer, i.e. the maximum number of characters that can be displayed when scrolling.
PK_LCD_TEXT_LINE_LENGTH (pg. 99)	The maximum number of characters that can be displayed on a single line of the LCD without scrolling.
PK_LCD_TEXT_LINES (pg. 99)	Maximum number of display lines in text mode.

16.4.1 PK_LCD_BITMAP_HEIGHT

Maximum number of vertical pixels in bitmap mode.

```
#define PK_LCD_BITMAP_HEIGHT 32
```

16.4.2 PK_LCD_BITMAP_WIDTH

Maximum number of horizontal pixels in bitmap mode.

```
#define PK_LCD_BITMAP_WIDTH 160
```

16.4.3 PK_LCD_BLINK_INTERVAL_DEFAULT

Default LCD blink interval in ms.

```
#define PK_LCD_BLINK_INTERVAL_DEFAULT 500 /* Default LCD blink interval in ms. */
```

16.4.4 PK_LCD_BLINK_INTERVAL_MAX

Maximum LCD blink interval in ms.

```
#define PK_LCD_BLINK_INTERVAL_MAX 5000 /* Maximum LCD blink interval in ms. */
```

16.4.5 PK_LCD_BLINK_INTERVAL_MIN

Minimum LCD blink interval in ms.

```
#define PK_LCD_BLINK_INTERVAL_MIN 100 /* Minimum LCD blink interval in ms. */
```

16.4.6 PK_LCD_BRIGHTNESS_0

The brightness of the LCD back lighting is 0 percent.

```
#define PK_LCD_BRIGHTNESS_0 0 /* The brightness of the LCD back lighting is 0 percent. */
```

16.4.7 PK_LCD_BRIGHTNESS_100

The brightness of the LCD back lighting is 100 percent.

```
#define PK_LCD_BRIGHTNESS_100 4 /* The brightness of the LCD back lighting is 100 percent. */
```

16.4.8 PK_LCD_BRIGHTNESS_25

The brightness of the LCD back lighting is 25 percent.

```
#define PK_LCD_BRIGHTNESS_25 1 /* The brightness of the LCD back lighting is 25 percent. */
```

16.4.9 PK_LCD_BRIGHTNESS_50

The brightness of the LCD back lighting is 50 percent.

```
#define PK_LCD_BRIGHTNESS_50 2 /* The brightness of the LCD back lighting is 50 percent. */
```

16.4.10 PK_LCD_BRIGHTNESS_75

The brightness of the LCD back lighting is 75 percent.

```
#define PK_LCD_BRIGHTNESS_75 3 /* The brightness of the LCD back lighting is 75 percent. */
```

16.4.11 PK_LCD_DISPLAY_MODE_BITMAP

The display mode of the LCD is bitmap mode.

```
#define PK_LCD_DISPLAY_MODE_BITMAP 0 /* The display mode of the LCD is bitmap mode. */
```

16.4.12 PK_LCD_DISPLAY_MODE_TEXT

The display mode of the LCD is text mode.

```
#define PK_LCD_DISPLAY_MODE_TEXT 1 /* The display mode of the LCD is text mode. */
```

16.4.13 PK_LCD_EVENT_MAX_NAME_LENGTH

Maximum length of an event name.

```
#define PK_LCD_EVENT_MAX_NAME_LENGTH 80
```

16.4.14 PK_LCD_ERROR_MAX_NAME_LENGTH

Maximum length of a status code name.

```
#define PK_LCD_ERROR_MAX_NAME_LENGTH 80
```

16.4.15 PK_LCD_ORIENTATION_NORMAL

The orientation of the LCD is normal.

```
#define PK_LCD_ORIENTATION_NORMAL 0 /* The orientation of the LCD is normal. */
```

16.4.16 PK_LCD_ORIENTATION_REVERSED

The orientation of the LCD is reversed.

```
#define PK_LCD_ORIENTATION_REVERSED 1 /* The orientation of the LCD is reversed. */
```

16.4.17 PK_LCD_REGION_FULL_SCREEN

Convenience definition for the full screen region

```
#define PK_LCD_REGION_FULL_SCREEN ((PK_LCD_TLCDRegion){0,0, PK_LCD_BITMAP_WIDTH,
PK_LCD_BITMAP_HEIGHT})
```

16.4.18 PK_LCD_SHIFT_INTERVAL_DEFAULT

Default LCD shift interval in ms.

```
#define PK_LCD_SHIFT_INTERVAL_DEFAULT 500 /* Default LCD shift interval in ms. */
```

16.4.19 PK_LCD_SHIFT_INTERVAL_MAX

Maximum LCD shift interval in ms.

```
#define PK_LCD_SHIFT_INTERVAL_MAX 5000 /* Maximum LCD shift interval in ms. */
```

16.4.20 PK_LCD_SHIFT_INTERVAL_MIN

Minimum LCD shift interval in ms.

```
#define PK_LCD_SHIFT_INTERVAL_MIN 100 /* Minimum LCD shift interval in ms. */
```

16.4.21 PK_LCD_TEXT_LINE_BUFFER_LENGTH

Number of characters in the LCD display line buffer, i.e. the maximum number of characters that can be displayed when scrolling.

```
#define PK_LCD_TEXT_LINE_BUFFER_LENGTH 40
```

16.4.22 PK_LCD_TEXT_LINE_LENGTH

The maximum number of characters that can be displayed on a single line of the LCD without scrolling.

```
#define PK_LCD_TEXT_LINE_LENGTH 20
```

16.4.23 PK_LCD_TEXT_LINES

Maximum number of display lines in text mode.

```
#define PK_LCD_TEXT_LINES 2
```

16.5 PK_LCD_DisableLogs

The PK_LCD_DisableLogs function disables debug logging in PK_LCD.

```
PK_STATUS PK_API PK_LCD_DisableLogs(
    IN TPikaHandle lcdHandle
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_SUCCESS	The function succeeded.

16.6 Errors

Macros

PK_LCD_ERROR_BASE_GENERAL (pg. 101)	General Error Codes
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The handle provided is not valid.
PK_LCD_ERROR_LCD_INVALID_BLINK_TIME (pg. 101)	The blinkTime parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid
PK_LCD_ERROR_LCD_INVALID_BRIGHTNESS (pg. 101)	The brightness parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid
PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)	The mode parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid
PK_LCD_ERROR_LCD_INVALID_LINE_NUMBER (pg. 101)	The lineNumber parameter passed in is invalid
PK_LCD_ERROR_LCD_INVALID_ORIENTATION (pg. 102)	The orientation parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid
PK_LCD_ERROR_LCD_INVALID_REGION_HORIZONTAL (pg. 102)	The horizontal region parameter passed in is invalid
PK_LCD_ERROR_LCD_INVALID_REGION_VERTICAL (pg. 102)	The vertical region parameter passed in is invalid
PK_LCD_ERROR_LCD_INVALID_SHIFT_TIME (pg. 102)	The shiftTime parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid

PK_LCD_ERROR_LCD_BLINK_WAS_NOT_SET (pg. 102)	The LCD blinking is not set
PK_LCD_ERROR_LCD_NOT_PRESENT (pg. 102)	The LCD is not present in the system
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	A NULL parameter was passed to the function requiring a non-NULL value.
PK_LCD_ERROR_OUT_OF_MEMORY (pg. 103)	There is insufficient memory available to execute the function.

16.6.1 PK_LCD_ERROR_BASE_GENERAL

General Error Codes

```
#define PK_LCD_ERROR_BASE_GENERAL 0x0000
```

16.6.2 PK_LCD_ERROR_DEVICE_INVALID_HANDLE

The handle provided is not valid.

```
#define PK_LCD_ERROR_DEVICE_INVALID_HANDLE (-0x2003)
```

16.6.3 PK_LCD_ERROR_LCD_INVALID_BLINK_TIME

The blinkTime parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_BLINK_TIME (-0x4404)
```

16.6.4 PK_LCD_ERROR_LCD_INVALID_BRIGHTNESS

The brightness parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_BRIGHTNESS (-0x4405)
```

16.6.5 PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE

The mode parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (-0x4402)
```

16.6.6 PK_LCD_ERROR_LCD_INVALID_LINE_NUMBER

The lineNumber parameter passed in is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_LINE_NUMBER (-0x4406)
```

16.6.7 PK_LCD_ERROR_LCD_INVALID_ORIENTATION

The orientation parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_ORIENTATION (-0x4401)
```

16.6.8 PK_LCD_ERROR_LCD_INVALID_REGION_HORIZONTAL

The horizontal region parameter passed in is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_REGION_HORIZONTAL (-0x4407)
```

16.6.9 PK_LCD_ERROR_LCD_INVALID_REGION_VERTICAL

The vertical region parameter passed in is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_REGION_VERTICAL (-0x4408)
```

16.6.10 PK_LCD_ERROR_LCD_INVALID_SHIFT_TIME

The shiftTime parameter in the **PK_LCD_TLCDConfig (pg. 93)** is invalid

```
#define PK_LCD_ERROR_LCD_INVALID_SHIFT_TIME (-0x4403)
```

16.6.11 PK_LCD_ERROR_LCD_BLINK_WAS_NOT_SET

The LCD blinking is not set

```
#define PK_LCD_ERROR_LCD_BLINK_WAS_NOT_SET (-0x440A)
```

16.6.12 PK_LCD_ERROR_LCD_NOT_PRESENT

The LCD is not present in the system

```
#define PK_LCD_ERROR_LCD_NOT_PRESENT (-0x4409)
```

16.6.13 PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED

A NULL parameter was passed to the function requiring a non-NULL value.

```
#define PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (-0x2001)
```

16.6.14 PK_LCD_ERROR_OUT_OF_MEMORY

There is insufficient memory available to execute the function.

```
#define PK_LCD_ERROR_OUT_OF_MEMORY (-0x2002)
```

16.7 PK_LCD_DisplayBitmap

The PK_LCD_DisplayBitmap function displays a bitmap on the LCD in the specified region.

```
PK_STATUS PK_API PK_LCD_DisplayBitmap(
    IN TPikaHandle lcdHandle,
    IN PK_LCD_TLCDRegion * region,
    IN PK_CHAR * bitmap
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
region	The pointer to the **PK_LCD_TLCDRegion (pg. 94)** defining the area to display the bitmap.
bitmap	The pointer to the buffer containing the context of the bitmap pattern.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)	The LCD is configured in text mode. This function should only be called in bitmap mode.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	Either the region or the bitmap parameter passed in is NULL.
PK_LCD_ERROR_LCD_INVALID_REGION_HORIZONTAL (pg. 102)	The sum of the horizontalPosition parameter and the horizontalWidth parameter in region is larger than **PK_LCD_BITMAP_WIDTH (pg. 96)**. The region is too big to be displayed.
PK_LCD_ERROR_LCD_INVALID_REGION_VERTICAL (pg. 102)	The sum of the verticalPosition parameter and the verticalLength parameter in region is larger than **PK_LCD_BITMAP_HEIGHT (pg. 96)**. The region is too big to be displayed.
PK_SUCCESS	The function succeeded.

Remarks

This function displays a bitmap on the LCD with the specified region parameter and bitmap pattern. It will

overwrite the original content in that region. If this function is called multiple times in succession and there are overlaps, the subsequent bitmaps will be always on the top.

Notes

This function can only be called in bitmap mode, otherwise, the error code **PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)** will be returned.

16.8 Events

Macros

PK_LCD_EVENT_LCD_BUTTON_PRESSED (pg. 104)	[Value: 5B00] The PK_LCD_EVENT_LCD_BUTTON_PRESSED event indicates that the button on the Appliance has been pressed.

16.8.1 PK_LCD_EVENT_LCD_BUTTON_PRESSED

[Value: 5B00] The PK_LCD_EVENT_LCD_BUTTON_PRESSED event indicates that the button on the Appliance has been pressed.

```
#define PK_LCD_EVENT_LCD_BUTTON_PRESSED 0x5B00
```

Parameters

Parameters	Description
P0	The number of times the button has been pressed continously, once at least.
P1	None.
P2	None.

Remarks

None.

16.9 PK_LCD_DisplayString

The PK_LCD_DisplayString displays a string to the specified line on the LCD.

```
PK_STATUS PK_API PK_LCD_DisplayString(
    IN TPikaHandle lcdHandle,
    IN PK_CHAR * string,
    IN PK_UINT length,
```

```
    IN PK_UINT lineNumber
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
string	The address of the pointer to the string of characters (0-255) to be displayed.
length	The length of the string to be displayed.
lineNumber	The index of the line on the LCD that the string will be displayed.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The string parameter passed in is NULL.
PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)	The LCD is configured in bitmap mode. It cannot display a string of characters in this mode.
PK_LCD_ERROR_LCD_INVALID_LINE_NUMBER (pg. 101)	The line number specified is too big. It should be less than the value of numOfLines within the **PK_LCD_TLCDInfo (pg. 93)** struct retrieved by **PK_LCD_GetInfo (pg. 108)**() function.
PK_SUCCESS	The function succeeded.

Remarks

This function displays a string of characters on the specified line of the LCD. Before the string is displayed, the line of the LCD will be cleared. If the length of the string is larger than **PK_LCD_TEXT_LINE_BUFFER_LENGTH (pg. 99)**, only the first **PK_LCD_TEXT_LINE_BUFFER_LENGTH (pg. 99)** of characters will be displayed. If the length of the string is larger than **PK_LCD_TEXT_LINE_LENGTH (pg. 99)**, the string will be scrolled across the LCD with the shift interval specified by the shiftTime parameter in the configure structure.

Notes

This function can only be called when the LCD is configured in the text mode. Otherwise, **PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)** error will be returned.

16.10 PK_LCD_EnableLogs

The PK_LCD_EnableLogs function enables debug logging in PK_LCD.

```
PK_STATUS PK_API PK_LCD_EnableLogs(
    IN TPikaHandle lcdHandle
```

```
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_SUCCESS	The function succeeded.

Remarks

All the logs go to a file named "pikalcdapilogs.txt" in the directory defined by PKH_LOGS_DIR system varible, with "/var/log/pika" as default value.

16.11 PK_LCD_ERROR_GetText

The PK_LCD_ERROR_GetText function returns the name of the status code in a user-provided buffer.

```
PK_CHAR *PK_API PK_LCD_ERROR_GetText(
    IN PK_STATUS status,
    IN PK_CHAR * buffer,
    IN PK_SIZE_T size
);
```

Parameters

Parameters	Description
status	The status code of the error to be retrieved.
buffer	The address of the buffer used to retrieve the status code name information.
size	The size of the buffer in bytes.

Return Values

Return Values	Description
buffer	The pointer to the buffer passed in by the caller.

Remarks

This function allows the user application to log PK_STATUS codes by name rather than an obscure id.

Notes

Use the **PK_LCD_ERROR_MAX_NAME_LENGTH (pg. 98)** constant when allocating the buffer to hold the status code name.

16.12 PK_LCD_EVENT_GetText

The PK_LCD_EVENT_GetText function returns the name of the event in a user -provided buffer.

```
PK_CHAR *PK_API PK_LCD_EVENT_GetText(
    IN PK_UINT eventId,
    IN PK_CHAR * buffer,
    IN PK_SIZE_T size
);
```

Parameters

Parameters	Description
eventId	The event id to be retrieved.
buffer	The address of the buffer used to retrieve the event name information.
size	The size of the buffer in bytes.

Return Values

Return Values	Description
buffer	The pointer to the buffer passed in by the caller.

Remarks

This function allows the user application to log incoming events by name rather than an obscure numeric value.

Notes

Use the **PK_LCD_EVENT_MAX_NAME_LENGTH (pg. 98)** constant when allocating the buffer to hold the event name.

16.13 PK_LCD_GetConfig

The PK_LCD_GetConfig function retrieves the current configuration settings of the specified LCD.

```
PK_STATUS PK_API PK_LCD_GetConfig(
    IN TPikaHandle lcdHandle,
    OUT PK_LCD_TLCDConfig * lcdConfig
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.

lcdConfig	The address of the **PK_LCD_TLCDConfig (pg. 93)** structure used to return the LCD configuration information.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The lcdConfig parameter passed in is NULL.
PK_SUCCESS	The function succeeded.

Remarks

This function retrieves the current LCD configuration settings.

User applications should call this function to initialize their **PK_LCD_TLCDConfig (pg. 93)** structure prior to using the **PK_LCD_SetConfig (pg. 111)** function. This ensures all parameters (including parameters added in later releases) are set to proper default values.

16.14 PK_LCD_GetInfo

The PK_LCD_GetInfo function retrieves the capability information of the specified LCD.

```
PK_STATUS PK_API PK_LCD_GetInfo(
    IN TPikaHandle lcdHandle,
    OUT PK_LCD_TLCDInfo * lcdInfo
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
lcdInfo	The address of the **PK_LCD_TLCDInfo (pg. 93)** type used to return the information about the LCD.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The lcdInfo parameter passed in is NULL.
PK_SUCCESS	The function succeeded.

Remarks

This function retrieves the capacity information of the LCD display, including the dimensions of the LCD in text mode and bitmap mode.

16.15 PK_LCD_Open

The PK_LCD_Open function allocates a LCD object and returns its handle to the calling routine.

```
PK_STATUS PK_API PK_LCD_Open(
    IN PK_VOID * reserved0,
    OUT TPikaHandle * lcdHandle
);
```

Parameters

Parameters	Description
reserved0	Reserved for future use. This field must be set to NULL.
lcdHandle	The address of the TPikaHandle type used to return the LCD handle.

Return Values

Return Values	Description
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The LCDHandle parameter passed in is NULL.
PK_LCD_ERROR_OUT_OF_MEMORY (pg. 103)	There is insufficient memory available to allocate resources for the LCD.
PK_LCD_ERROR_LCD_NOT_PRESENT (pg. 102)	The LCD is not currently present in the system.
PK_SUCCESS	The function succeeded.

Remarks

This function allocates all the resources required to operate the LCD and initiates communication with the LCD driver. A handle is returned to be used for all further function invocations dealing with the LCD. Later calls to PK_LCD_Open will increase the use count, there will be only one instance of LCD service running in the system.

16.16 PK_LCD_ResetEventHandler

The PK_LCD_ResetEventHandler function reset the callback function used to notify events generated by LCD.

```
PK_STATUS PK_API PK_LCD_ResetEventHandler(
    IN TPikaHandle lcdHandle,
    IN PK_LCD_Callback callback
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
callback	The function pointer to the callback function.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The callback parameter passed in is NULL.
PK_SUCCESS	The function succeeded.

16.17 PK_LCD_SetBlink

The PK_LCD_SetBlink function will set a region in the LCD to blink with the speed specified by the displayBlinkTime parameter in the configure struct.

```
PK_STATUS PK_API PK_LCD_SetBlink(
    IN TPikaHandle lcdHandle,
    IN PK_LCD_TLCDRegion * region
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
region	pointer to the **PK_LCD_TLCDRegion (pg. 94)** that defines the area set to blink.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)	The LCD is configured in the text mode. This function should only be called in the bitmap mode.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The region parameter passed in is NULL.
PK_LCD_ERROR_LCD_INVALID_REGION_HORIZONTAL (pg. 102)	The sum of the horizontalPosition parameter and the horizontalWidth parameter in region is larger than the maximum number of horizontal pixels. The region is too big to be displayed. The maximum number of horizontal pixels can be retrieved by **PK_LCD_GetInfo (pg. 108)**().

PK_LCD_ERROR_LCD_INVALID_REGION_VERTICAL (pg. 102)	The sum of the verticalPosition parameter and the verticalLength parameter in region is larger than the maximum number of vertical pixels. The region is too big to be displayed.The maximum number of vertical pixels can be retrieved by **PK_LCD_GetInfo (pg. 108)**().
PK_SUCCESS	The function succeeded.

Remarks

This function sets an area in the LCD with the specified region parameter to blink. It will overwrite the setBlink functions in the same region. The blink action in a region will continue until the UnsetBlink function is called with the same region parameter, the display mode of the display is changed or the **PK_LCD_Clear (pg. 92)** function is call.

Notes

This function can only be called in bitmap mode, otherwise the error code **PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)** will be returned.

16.18 PK_LCD_SetConfig

The PK_LCD_SetConfig function sets the configuration settings of the specified LCD.

```
PK_STATUS PK_API PK_LCD_SetConfig(
    IN TPikaHandle lcdHandle,
    IN PK_LCD_TLCDConfig * lcdConfig
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
lcdConfig	The address of the **PK_LCD_TLCDConfig (pg. 93)** structure containing the information used to configure the LCD.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to an LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The lcdConfig parameter passed in is NULL.
PK_LCD_ERROR_LCD_INVALID_ORIENTATION (pg. 102)	The orientation parameter in the lcdConfig is invalid.
PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)	The mode parameter specified in the lcdConfig is invalid.
PK_LCD_ERROR_LCD_INVALID_SHIFT_TIME (pg. 102)	The shiftTime parameter specified in the lcdConfig is out of range.

PK_LCD_ERROR_LCD_INVALID_BLINK_TIME (pg. 101)	The blinkTime parameter specified in the lcdConfig is out of range.
PK_LCD_ERROR_LCD_INVALID_BRIGHTNESS (pg. 101)	The brightness parameter specified in the lcdConfig is not valid, it should be one of the PK_LCD_BRIGHTNESS_x values defined in this header file.
PK_SUCCESS	The function succeeded.

Remarks

This function configures the LCD with the settings provided in the lcdConfig struct pointer. The settings become effective immediately after the function call. If the mode of the display is changed, all the content and operations, such as blinking and shifting associated with previous mode will be cleared.

User applications should initialize the **PK_LCD_TLCDConfig (pg. 93)** structure by calling **PK_LCD_GetConfig (pg. 107)** and passing in the lcdConfig structure prior to calling this function. This guarantees that all fields (including fields added in later releases of this software) are set to proper default values.

16.19 PK_LCD_SetEventHandler

The PK_LCD_SetEventHandler function sets the callback function for notifying events generated by LCD.

```
PK_STATUS PK_API PK_LCD_SetEventHandler(
    IN TPikaHandle lcdHandle,
    IN PK_LCD_Callback callback,
    PK_VOID * userData
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
callback	The function pointer to the callback function.
userData	The user data that will be reported back to the user in the events.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The callback parameter passed in is NULL.
PK_SUCCESS	The function succeeded.

Remarks

Only one callback can be registered at a time. The callback function registered later will replace the one

registered earlier. When the callback is not needed anymore, an **PK_LCD_ResetEventHandler (pg. 109)** should be called to reset the callback function.

16.20 PK_LCD_SetFontLib

The PK_LCD_SetFontLib function sets the bitmap library of the ASCII characters to replace to the default one.

```
PK_STATUS PK_API PK_LCD_SetFontLib(
    IN TPikaHandle lcdHandle,
    IN PK_CHAR * fontLib
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
fontLib	The pointer to the font library.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The fontLib parameter passed in is NULL.
PK_SUCCESS	The function succeeded.

Remarks

The size of the font's bitmap is set to 5x7. With a 1 pixel margin, the size of the bitmap for each font is 6x8. As a result, the bitmap pattern of each character is 6 bytes long with each byte representing the bitmask of the corresponding column. There are total 256 spaces in the library, hence the size of the total library is 256*6 = 1536 bytes.

16.21 PK_LCD_UnsetBlink

The PK_LCD_UnsetBlink function will cancel the blink operation previously set by the **PK_LCD_SetBlink (pg. 110)** function call.

```
PK_STATUS PK_API PK_LCD_UnsetBlink(
    IN TPikaHandle lcdHandle,
    IN PK_LCD_TLCDRegion * region
);
```

Parameters

Parameters	Description
lcdHandle	The LCD handle returned by **PK_LCD_Open (pg. 109)**.
region	pointer to the **PK_LCD_TLCDRegion (pg. 94)** defines the area to unset the blinking.

Return Values

Return Values	Description
PK_LCD_ERROR_DEVICE_INVALID_HANDLE (pg. 101)	The lcdHandle parameter does not refer to the LCD previously opened with the **PK_LCD_Open (pg. 109)** function.
PK_LCD_ERROR_NULL_PARAMETER_SPECIFIED (pg. 102)	The region parameter passed in is NULL.
PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)	The LCD is configured in text mode. This function should only be called in the bitmap mode.
PK_LCD_ERROR_LCD_INVALID_REGION_HORIZONTAL (pg. 102)	The sum of the horizontalPosition parameter and the horizontalWidth parameter in region is larger than the maximum number of horizontal pixels. The region is too big to be displayed. The maximum number of horizontal pixels can be retrieved by **PK_LCD_GetInfo (pg. 108)**().
PK_LCD_ERROR_LCD_INVALID_REGION_VERTICAL (pg. 102)	The sum of the verticalPosition parameter and the verticalLength parameter in region is larger than the maximum number of vertical pixels. The region is too big to be displayed. The maximum number of vertical pixels can be retrieved by **PK_LCD_GetInfo (pg. 108)**().
PK_LCD_ERROR_LCD_BLINK_WAS_NOT_SET (pg. 102)	There is no blink operation set on the region specified by the region parameter.
PK_SUCCESS	The function succeeded.

Remarks

None.

Notes

This function can only be called in bitmap mode, otherwise the error code **PK_LCD_ERROR_LCD_INVALID_DISPLAY_MODE (pg. 101)** will be returned.

17 Appendix B - Timezone Codes

The following table lists time zone codes for most countries.

Countries	Current Time Zone Notation
Afghanistan	UCT-4:30
Albania	MET-1METDST
Algeria	UCT-1
American Samoa	UCT11
Andorra	MET-1METDST
Angola	UCT-1
Anguilla	UCT4
Antigua and Barbuda	UCT4
Argentina	SAT3
Armenia	UCT-4
Aruba	UCT4
Australia (Broken Hill and South Australia)	CST-9:30CDT
Australia (Lord Howe Island)	LHT-10:30LHDT
Australia (New South Wales, Capitol Territory, Victoria)	EST-10EDT
Australia (Northern Territory)	UCT-9:30
Australia (Queensland)	UCT-10
Australia (Tasmania)	TST-10TDT
Australia (Western)	UCT-8
Austria	MEZ-1MESZ
Azerbaijan	UCT-3
Bahamas	EST5EDT
Bahrain	UCT-3
Bangladesh	UCT-6
Barbados	UCT4
Belarus	EET-2EETDST

Belgium	MET-1METDST
Belize	UCT6
Benin	UCT-1
Bermuda	AST4ADT
Bhutan	UCT-6
Bolivia	UCT4
Bonaire	UCT4
Bosnia Herzogovina	MET-1METDST
Botswana	UCT-2
Brazil (East, including All Coast and Brasilia)	EBST3EBDT
Brazil (Fernando de Noronha)	NORO2
Brazil (Trinity of Acre)	ACRE5
Brazil (West)	WBST4WBDT
British Virgin Islands	UCT4
Brunei	UCT-8
Bulgaria	EET-2EETDST
Burkina Faso	UCT
Burma	UCT-6:30
Burundi	UCT-2
Cambodia	UCT-7
Cameroon	UCT-1
Canada (Atlantic)	AST4ADT
Canada (Central)	CST6CDT
Canada (Eastern)	EST5EDT
Canada (Mountain)	MST7MDT
Canada (Newfoundland)	NST3:30NDT
Canada (Pacific and Yukon)	PST8PDT
Cape Verde	UCT1
Cayman Islands	UCT5
Central African Republic	UCT-1

Chad	UCT-1
Chile	CST4CDT
Chile (Easter Island)	EIST6EIDT
China	CST-8
Christmas Islands	UCT-7
Cocos (Keeling) Islands	UCT-6:30
Colombia	UCT5
Congo	UCT-1
Cook Islands	UCT10
Costa Rica	UCT6
Cote d'Ivoire	UCT
Croatia	MET-1METDST
Cuba	UCT5
Curacao	UCT4
Cyprus	EET-2EETDST
Czech Republic	MET-1METDST
Denmark	MET-1METDST
Djibouti	UCT-3
Dominica	UCT4
The Dominican Republic	UCT4
Ecuador	UCT5
Ecuador (Galapagos Islands)	UCT6
Egypt	EST-2EDT
El Salvador	UCT6
Equatorial Guinea	UCT-1
Eritrea	UCT-3
Estonia	EET-2EETDST
Ethiopia	UCT-3
Faroe Islands	WET0WETDST
Fiji	UCT-12

Finland	EET-2EETDST
France	MET-1METDST
French Guiana	SAT3
French Polynesia	UCT10
Gabon	UCT-1
The Gambia	UCT
Georgia	EUT-4EUTDST
Germany	MEZ-1MESZ
Ghana	UCT
Gibraltar	MET-1METDST
Greece	EET-2EETDST
Greenland (Scorsbysund)	EUT1EUTDST
Greenland (Thule)	AST4ADT
Grenada	UCT4
Guadeloupe	UCT4
Guam	UCT-10
Guatemala	UCT6
Guinea Bissau	UCT
Guyana	UCT3
Haiti	EST5EDT
Hawaii	UCT10
Honduras	UCT6
Hong Kong	UCT-8
Hungary	MET-1METDST
Iceland	UCT
India	UCT-5:30
Indonesia (Central)	UCT-8
Indonesia (East)	UCT-9
Indonesia (West)	UCT-7
Iran	UCT-3:30

Iraq	IST-3IDT
Ireland	GMT0BST
Israel	IST-2IDT
Italy	MET-1METDST
Jamaica	UCT5
Japan	JST
Johnston Islands	UCT10
Jordan	JST-2JDT
Juan Fernandez Islands	UCT5
Kazakhstan	EUT-6EUTDST
Kenya	UCT-3
Kiribati	UCT-12
Kuwait	UCT-3
Kyrgyzstan	UCT-5
Laos	UCT-7
Latvia	EET-2EETDST
Lebanon	EUT-2EUTDST
Lesotho	UCT-2
Liberia	UCT
Libya	UCT-2
Liechtenstein	MET-1METDST
Lithuania	EET-2EETDST
Luxembourg	MET-1METDST
Macao	UCT-8
Macedonia	MET-1METDST
Madagascar	UCT-3
Malawi	UCT-2
Malaysia	MST-8
Maldives	UCT-5
Mali	UCT

Malta	MET-1METDST
Mariana Islands	UCT-10
Martinique	UCT4
Mauritania	UCT
Mauritius	UCT-4
Mayotte	UCT-3
Mexico	CST6CDT
Mexico (Baja N.)	PST8PDT
Mexico (Baja S.)	MST7MDT
Midway Islands	UCT11
Moldova	EET-2EETDST
Monaco	MET-1METDST
Mongolia	EUT-8EUTDST
Montenegro	MET-1METDST
Montserrat	UCT4
Morocco	UCT
Mozambique	UCT-2
Namibia	UCT-2
Nauru	UCT-12
Nepal	UCT-5:45
The Netherlands Antilles	UCT4
The Netherlands	MET-1METDST
New Caledonia	UCT-11
New Hebrides	UCT-11
New Zealand	NZST-12NZDT
New Zealand (Chatham Island)	CIST-12:45CIDT
Nicaragua	UCT6
Niger	UCT-1
Nigeria	UCT-1
Niue Islands	UCT11

Norfolk Island	UCT-11:30
North Korea	KST
Norway	MET-1METDST
Oman	UCT-4
Pakistan	UCT-5
Palau	UCT-9
Panama	UCT5
Papua New Guinea	UCT-10
Paraguay	UCT4
Peru	UCT5
Philippines	UCT-8
Pitcairn Island	UCT-9
Poland	MET-1METDST
Portugal	PWT0PST
Portugal (Azores)	EUT1EUTDST
Puerto Rico	UCT4
Qatar	UCT-3
Reunion	UCT-4
Romania	EET-2EETDST
Russia (Moscow)	MST-3MDT
Russian Fed. Zone 1 (Kaliningrad)	RFT-2RFTDST
Russian Fed. Zone 10 (Magadan)	RFT-11RFTDST
Russian Fed. Zone 11 (Petropavolovsk-Kamchatsky)	RFT-12RFTDST
Russian Fed. Zone 2 (St. Petersburg)	RFT-3RFTDST
Russian Fed. Zone 3 (Izhevsk)	RFT-4RFTDST
Russian Fed. Zone 4 (Ekaterinburg)	RFT-5RFTDST
Russian Fed. Zone 5 ((Novosibirsk)	RFT-6RFTDST
Russian Fed. Zone 6 (Krasnojarsk)	RFT-7RFTDST
Russian Fed. Zone 7 ((Irkutsk)	RFT-8RFTDST
Russian Fed. Zone 8 (Yakatsk)	RFT-9RFTDST

Russian Fed. Zone 9 (Vladivostok)	RFT-10RFTDST
Rwanda	UCT-2
Saint Pierre & Miquelon	NAST3NADT
San Marino	MET-1METDST
Sao Tome and Principe	UCT
Saudi Arabia	UCT-3
Senegal Sierra Leone	UCT
Serbia	MET-1METDST
The Seychelles	UCT-4
Singapore	UCT-8
Slovakia	MET-1METDST
Slovenia	MET-1METDST
Solomon Islands	UCT-11
Somalia	UCT-3
South Africa	SAST-2
South Georgia	UCT3
South Korea	KST
Spain	MET-1METDST
Spain (Canary Islands)	WET0WETDST
Sri Lanka	UCT-5:30
St. Helena	UCT
St. Kitts-Nevis	UCT4
St. Lucia	UCT4
St. Vincent and the Grenadines	UCT4
Sudan	UCT-2
Suriname	UCT3
Swaziland	UCT-2
Sweden	MET-1METDST
Switzerland	MEZ-1MESZ
Syria	SST-2SDT

Tahiti	UCT10
Taiwan	UCT-8
Tajikistan	UCT-5
Tanzania	UCT-3
Thailand	UCT-7
Togo	UCT
Tonga	UCT-13
Trinidad and Tobago	TTST4
Tunisia	UCT-1
Turkey	EET-2EETDST
Turkmenistan	UCT-5
Turks and Caicos Islands	EST5EDT
Tuvalu	UCT-12
Uganda	UCT-3
Ukraine	EET-2EETDST
Ukraine (Simferopol)	EUT-3EUTDST
United Arab Emirates	UAEST-4
United Kingdom	GMT0BST
Uruguay	SAT3
US Virgin Islands	UCT4
USA (Alaska)	NAST9NADT
USA (Aleutian Islands)	AST10ADT
USA (Arizona)	MST7
USA (Central)	CST6CDT
USA (Eastern)	EST5EDT
USA (Indiana)	EST5
USA (Mountain)	MST7MDT
USA (Pacific)	PST8PDT
Uzbekistan	UCT-5
Vanuatu	UCT-11

Vatican City	MET-1METDST
Venezuela	UCT4
Vietnam	UCT-7
Wake Islands	UCT-12
Wallis and Futana Islands	UCT-12
Western Samoa	UCT11
Yemen	UCT-3
Zaire (Kasai)	UCT-2
Zaire (Kinshasa)	UCT-1
Zambia	UCT-2
Zimbabwe	UCT-2

Index

M

N

P

R

S

T

U

V

W